Санкт-Петербургский государственный университет
Кафедра теории систем управления электрофизической аппаратурой

Ращенко Юлия Владимировна

Магистерская диссертация

Применение модифицированных искусственных нейронных сетей в задачах машинного зрения

Направление 010900
«Прикладные математика и физика»
Магистерская программа «Информационные и ядерные технологии»

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Козынченко В. А.

Санкт-Петербург
2018

Оглавление

Введение

Искусственные нейронные сети (ИНС) являются перспективной областью для исследования. Они позволяют сходным образом решать совершенно разные прикладные задачи, поэтому находят применение в широком спектре областей: медицина, экономика, робототехника и др. При этом в ряде задач нейросети оказываются эффективнее других математических алгоритмов [1]. Важное значение в этом успехе нейросетей играет их способность к обучению, что позволяет выявлять сложные зависимости в поступающей на вход информации, производить обобщение. При этом нейронные сети нечувствительны к незначительным изменениям входных образов, шумам и искажениям входной информации, что позволяет им легко адаптироваться в условиях изменяющейся внешней среды [2].

Отмеченное поведение искусственных нейронных сетей в некотором роде напоминает мыслительную деятельность человека, поэтому ИНС являются также одним из способов моделирования искусственного интеллекта.

При решении сложных прикладных задач все чаще встает необходимость в использовании глубокого обучения, требующего больших вычислительных ресурсов, однако важным преимуществом нейросетевого подхода является возможность эффективного распараллеливания алгоритма. Поэтому успех в нейросетевой области также коррелирует с развитием технологий ускоренных вычислений. Особенно сильный толчок обеспечила архитектура параллельных вычислений CUDA от NVIDIA, которая позволяет существенно увеличить вычислительную производительность благодаря использованию GPU (графических процессоров) [3].

Важно отметить, что исследования в области искусственного интеллекта и искусственных нейронных сетей проводятся исследовательскими группами по всему миру, включая такие крупные компании как Google [4], Yandex [5], Facebook [6] и другие. Если говорить о задачах машинного зрения, то в данной области за последние годы были получены весьма впечатляющие результаты. Ниже перечислены некоторые из таких задач и

соответственно результаты, полученные при их решении:

1. генерация текстур [1];
2. перенос стиля [7];
3. сегментация объектов [8];
4. преобразование изображения в изображение [9] (см. рис. 1);
5. классификация [10].

Многие из перечисленных результатов основаны на применении сверточных нейронные сетей. Помимо этого разработаны новые интересные архитектурные решения, направленные на устранение различных проблем и улучшение качества глубокого обучения:

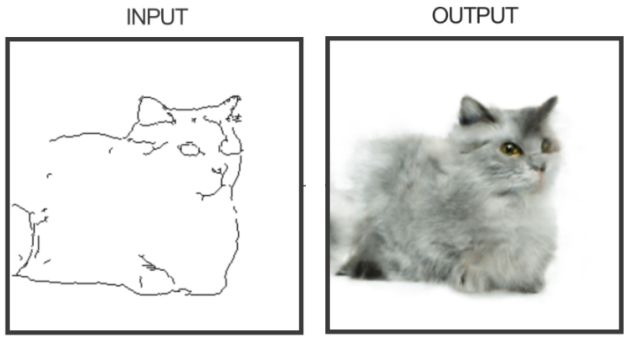

Рис. 1. Пример преобразования скетча в фотографию. Изображение взято с сайта https://affinelayer.com/pixsrv.

1. каскадные нейронные сети СССР [11];
2. генеративно-соревновательные нейронные сети GAN — использование одной нейросети для обучения другой, но для обучения требуются размеченные данные (обучение с учителем) [9];
3. нейросети CycleGAN — продолжение идеи GAN, но обучение без учителя [12];
4. остаточные нейросети ResNet [13];
5. нейросети Inception: новый подход к организации нейронных сетей, заключающийся в обработке входа различными методами и конкатенации результатов этой обработки [14].

Искусственный нейрон, искусственная нейронная сеть

Универсальность искусственных нейронных сетей заключается в том, что сложный алгоритм работы любой нейросетевой архитектуры строится из множества простых однотипных действий, реализуемых элементарными вычислительными узлами — искусственными нейронами. Функционирование искусственного нейрона в той или иной степени сходно с деятельностью его прототипа — биологического нейрона, являющегося основным структурным и функциональным элементом нервной системы животных. Упрощенная схема биологического нейрона представлена на рис. 2. Передача сигналов между нейронами происходит с помощью электрохимических импульсов. Они принимаются в синапсах и при определенных условиях могут возбудить нейрон. В этом случае нейрон посылает сигнал по аксону, который разветвляется на дендриты, ведущие к другим нейронам. Нервные клетки расположены по всему телу животного, и именно они управляют всеми процессами в теле животного, начиная от пищеварительной системы и заканчивая мыслительной деятельностью человека [15].

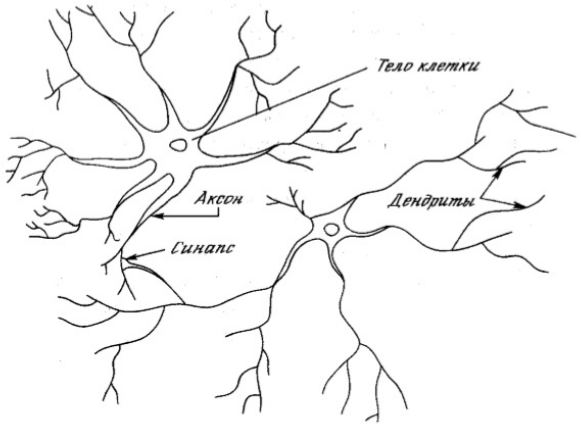

Рис. 2. Две взаимодействующие нервные клетки [2].

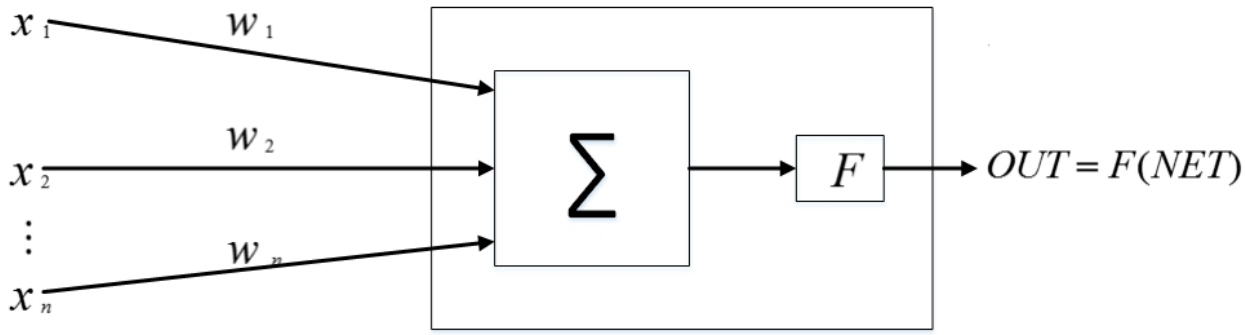

Рис. 3. Модель искусственного нейрона [2].

Существуют различные математические модели искусственных нейронов, имитирующие работу нервных клеток, но в основном применяются

вычислительные узлы, принцип функционирования которых можно представить схемой, изображенной на рис. 3. Основой такой модели является нейрон Мак-Каллока—Питса [16]. На входы искусственного нейрона поступает множество сигналов $x_1, \dots x_n$. Каждый из них умножается на вес $w_i, \quad i = \overline{1,n}$, соответствующий входу i, и результаты умножения суммируются. К полученной сумме применяется активационная функция F, и выход функции идет на вход другим нейронам.

В качестве активационных функций могут применяться различные правила, далее приведены некоторые из них, они будут впоследствии использоваться в данной работе:

1. Linear — линейная функция:

$$f(y) = y; \tag{1}$$

2. ReLU — «rectified linear unit»:

$$f(y) = \begin{cases} y, \text{ если } y > 0, \\ 0, \text{иначе;} \end{cases} \tag{2}$$

3. Sigmoid — сигмоидальная функция:

$$f(y) = \frac{1}{1 + e^{-y}}; \tag{3}$$

4. Sigmoid_Antisimmetric — антисимметричная сигмоидальная функция:

$$f(y) = \frac{2}{1 + e^{-y}} - 1. \tag{4}$$

5. Softmax:

$$f(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{l} e^{y_j}}, \quad i = \overline{1,l}, \tag{5}$$

где l — количество нейронов слоя.

6. Hardmax:

$$f(y_i) = \begin{cases} 1, \text{ если } y_i = \max_j y_j, \quad j = \overline{1,l}, \\ 0, \text{ в противном случае,} \end{cases} \tag{6}$$

где l — количество нейронов слоя.

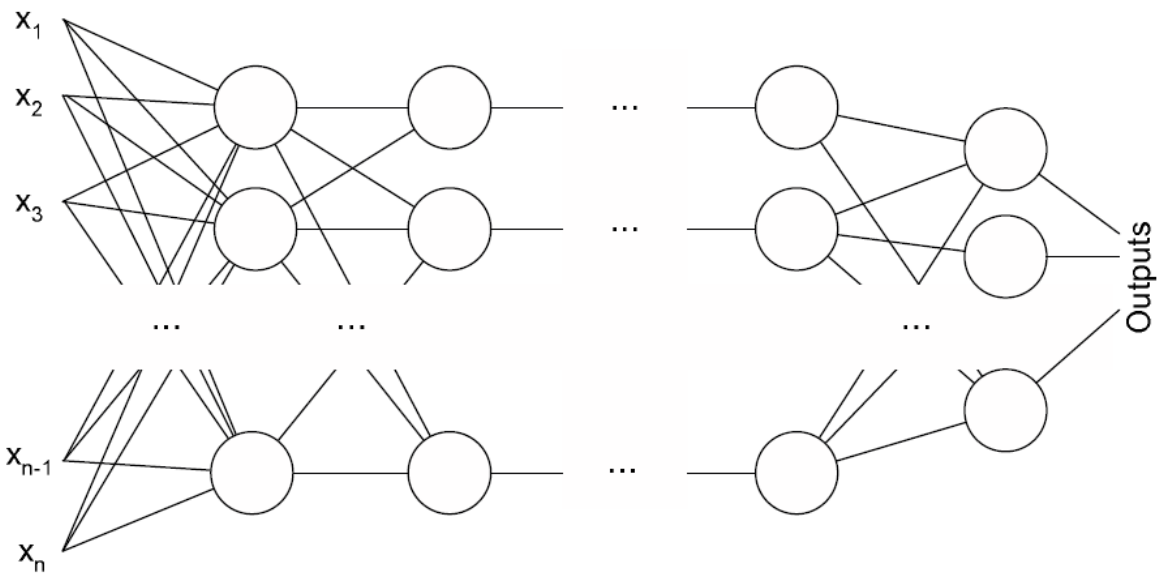

Рис. 4. Послойная архитектура искусственной нейронной сети [17].

Искусственные нейроны объединяют в нейронные сети. Обычно применяются так называемые послойные архитектуры как на рис. 4, когда нейроны объединяются в группы (слои), и связи устанавливаются только между нейронами соседних слоев. Такая архитектура удобна тем, что производимые в слое вычисления можно представить в виде матричного умножения, что дает больше возможностей для ускорения вычислений. Однако такая архитектура довольно плохо соответствует принципам соединения нейронов в нервной системе животных, и поэтому параллельно также ведется работа по исследованию так называемых хаотических нейронных сетей [18], хотя в этой области на данный момент получены гораздо менее впечатляющие результаты.

Обучение искусственной нейронной сети

Важным свойством искусственных нейронных сетей является их способность к обучению, то есть модификации своего поведения в ответ на

изменение внешней среды. В результате обучения нейросеть самонастраивается для обеспечения требуемой реакции в зависимости от входных данных. Технически обучение — это изменение весов $w_1, \ldots, w_n$ на рис. 3. Различают обучение с учителем и без учителя [2].

Обучение с учителем предполагает наличие обучающих пар, состоящих из входного паттерна и соответствующего ему ожидаемого отклика нейросети. Нейронная сеть обрабатывает входной вектор и выдает некоторый ответ. Этот ответ сравнивается со второй частью обучающей пары — ожидаемым выходом, вычисляется разница, или ошибка, и в соответствии с ней весовые коэффициенты нейронов изменяются по некоторому алгоритму, стремящемуся уменьшить ошибку. Целью обучения является достижение ошибки некоторого допустимого уровня, который устанавливается, исходя из требований задачи.

При обучении без учителя нейронная сеть располагает только множеством входных паттернов. В процессе обучения нейросеть выявляет различные зависимости и корреляции во входном множестве и в соответствии с ними разделяет это множество на кластеры. При предъявлении нейросети экземпляра, принадлежащего некоторому кластеру, она выдает в результате вектор, характеризующий данный кластер.

Можно заметить, что обучение с учителем требует предварительного процесса подготовки обучающего множества, который иногда может оказаться трудоемким. Поэтому наиболее привлекательным представляется обучение без учителя, и работы в этом направлении активно ведутся [12]. Однако на данный момент в некоторых задачах такое обучение невозможно, либо дает менее качественные или ненадежные результаты.

Виды искусственных нейронных сетей

Существуют различные парадигмы искусственных нейронных сетей, отличающихся математической моделью искусственного нейрона, способом организации нейронов, методами обучения. Наибольшее практическое применение находят искусственные нейронные сети, разработанные на основе

многослойного персептрона [19, 20] и сверточной нейронной сети [21, 22, 10]. Также проводятся научные исследования в области адаптивной резонансной теории [23, 24, 25, 26].

Многослойный персептрон (рис. 4) состоит из нескольких слоев нейронов Мак-Каллока—Питтса. Персептрон является полносвязной искусственной нейронной сетью: это значит, что нейроны каждого слоя связаны со всеми нейронами соседних слоев. Основу современного персептрона заложил американский нейрофизиолог Фрэнк Розенблатт, который в 1958 году практически реализовал сеть, предложенную ранее У. Мак-Каллоком и У. Питтсом, в виде компьютерной программы, а впоследствии и как электронное устройство [19]. Первоначальная версия персептрона, как и нейрон Мак-Каллока—Питтса, могла работать только с бинарными входами, однако впоследствии его область применения была расширена и на непрерывные сигналы. Многослойный персептрон находит практическое применение, например, в задачах классификации и аппроксимации. Также полносвязные слои персептрона используются в архитектурах других нейронных сетей.

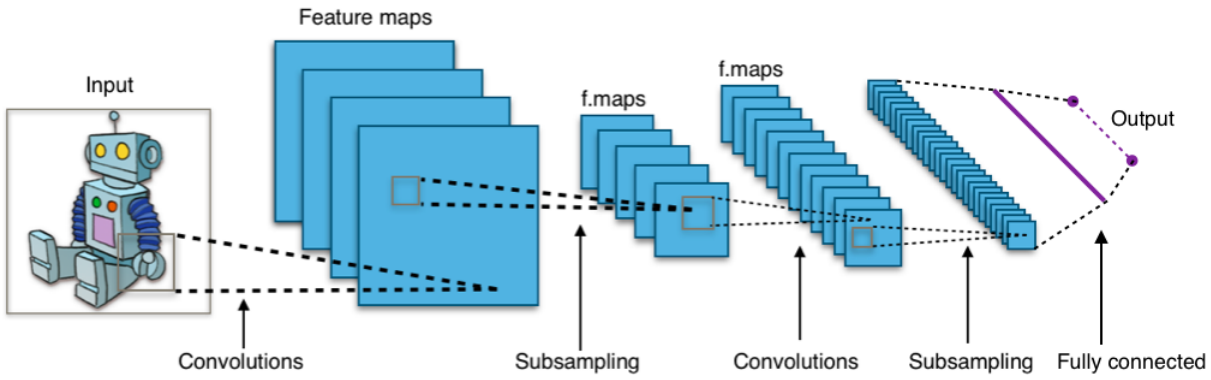

Рис. 5. Сверточная нейронная сеть [27].

Сверточная нейронная сеть [21] (рис. 5) является развитием многослойной полносвязной нейронной сети и нацелена на эффективное распознавание изображений. Данная архитектура была предложена Яном Лекуном в 1988 году и вдохновлена нейробиогическими исследованиями в области зрительной коры. Ключевой особенностью и отличием сверточной нейросети от персептрона является то, что нейроны слоя имеют не индивидуальные весовые коэффициенты, а используют разделенные веса: матри-

цы весов небольшого размера, которые также называют ядрами свертки. Таким образом, сверточная сеть имеет гораздо меньшее количество настраиваемых параметров по сравнению с полносвязной сетью, из чего следует более высокая производительность и экономичность по используемой памяти. Сверточные нейронные сети являются мощным инструментом в задачах машинного зрения.

Методы обучения искусственных нейронных сетей

Алгоритм изменения весовых коэффициентов — это важный аспект искусственных нейронных сетей, его можно варьировать для достижения желаемых результатов в некоторой задаче. Наиболее общепринятым и повсеместно применяемым алгоритмом обучения является градиентная оптимизация с обратным распространением ошибки [28]. Метод является модификацией классического метода градиентного спуска. Основная идея обратного распространения состоит в распространении сигналов ошибки от выходов сети к ее входам, то есть в направлении, обратном прямому распространению сигналов в обычном режиме. Таким образом, можно сказать, что при изменении весов нейронов используется информация об ошибке нейронов всех последующих слоев. Важным условием применения данного алгоритма является дифференцируемость активационной функции нейронов.

Проводятся также параллельная разработка и исследование других алгоритмов обучения нейросетей, хотя такие исследования носят сейчас скорее теоретический характер. Одним из таких алгоритмов является правило обучения Хебба [29]. Оно названо в честь канадского нейрофизиолога Дональда Хебба, который в 1949 году сформулировал физиологическое правило синаптической модификации, или постулат обучения Хебба [30]. Основная его идея заключается в том, что эффективность синапса между двумя нейронами повышается при многократной активации этих нейронов через данный синапс. Таким образом, обучение по правилу Хебба является в высшей степени локальным: вес нейросети изменяется под воздействием корреляций нейронов, которые этот вес соединяет. Также следует от-

метить, что хеббоподобные правила обучения — это самоорганизующиеся алгоритмы, реализующие обучение без учителя.

Постановка задачи

На данный момент большинство сверточных нейронных сетей обучаются методом градиентного спуска с обратным распространением ошибки. Такие нейросети достигают впечатляющих результатов (см. раздел «Введение»). Однако существует ряд проблем, которые могут возникать при данном методе обучения, например:

- проблема затухающих градиентов: появляется при обучении глубоких нейронных сетей и заключается в маленьких значениях градиента функционала качества на слоях, отдаленных от выхода, что затрудняет сходимость алгоритма обучения [13];
- попадание в локальный минимум: проблема, известная при использовании градиентной оптимизации; поверхность ошибки, на которой ищется минимум, сильно изрезана, и при попадании на «плато» или в «овраг» затруднительно выбраться оттуда и продолжить оптимизацию [2];
- переобучение: обычно возникает при обучении глубоких нейросетей и недостаточной обучающей выборке; проявляет себя большими значениями весов, маленькой ошибкой обучения, но большой ошибкой при тестировании сети на данных, не участвующих в обучении, что говорит о пониженной обобщающей способности модели [31];
- деградация качества обучения: наблюдается при углублении нейросетевой архитектуры, когда добавление новых слоев не улучшает качество нейросети, а даже увеличивает ошибку, полученную при обучении менее глубокого аналога [13];

Существуют различные подходы, направленные на устранение или уменьшение некоторых из этих проблем без отказа от обратного распро-

странения ошибки [13, 32, 31]. Однако также представляет интерес разработка альтернативных механизмов обучения.

Таким, образом, задача данной научно-исследовательской работы — разработка модифицированных методов обучения сверточных нейронных сетей, основанных на постулате Хебба, и оценка их качества и работоспособности на примере задачи машинного зрения. При этом будут проведены эксперименты с различными вариациями правила Хебба, что позволит выявить наиболее успешную конфигурацию алгоритма обучения. Также будет проведено сравнительное тестирование нейросетей с одинаковыми архитектурой и обучающим множеством, но разными алгоритмами обучения: градиентным с обратным распространением и хеббовским. Значение функционала качества (будет введен далее) на этапе тестирования, а также продолжительность обучения будут являться критериями оценки исследуемых моделей.

Обзор литературы

При написании данной магистерской диссертации использована учебная, научная и научно-популярная литература, статьи в периодических изданиях Российской Федерации и других стран, труды научных конференций, а также электронные ресурсы.

Основными источниками освоения базовых принципов искусственных нейронных сетей явились научные и научно-популярные издания: Уоссермен Ф. «Нейрокомпьютерная техника» [2], Хайкин С. «Нейронные сети: полный курс» [29], Осовский С. «Нейронные сети для обработки информации» [33]. При этом [2] особенно полезен на начальных этапах работы в данной области, так как дает довольно поверхностный, но доступный обзор основ ИНС. Градиентное обучение многослойного персептрона рассмотрено с опорой на материалы книги [33]. Метод главных компонент и обучение Хебба изучены в основном по изданию [29]. Также для понимания физиологических механизмов, лежащих в основе моделирования ИНС, использовалось учебное пособие Дубынин В. А. «Регуляторные системы ор-

ганизма человека» [15] и пройден онлайн-курс «Физиология центральной нервной системы» от Московского государственного университета на сайте www.openedu.ru.

Самые современные идеи, подходы и результаты в нейросетевой сфере были изучены на примере статей, опубликованных за последние 4 года в трудах таких известных для специалистов по компьютерному зрению международных конференций, как Conference on Computer Vision and Pattern Recognition, International Conference on Computer Vision и других. Это работы Gatys L. A. et al. «Texture synthesis using convolutional neural networks» [1], Luan F. et al. «Deep photo style transfer» [7], Zheng S. et al. «Conditional Random Fields as Recurrent Neural Networks» [8], Isola P. et al. «Image-to-Image Translation with Conditional Adversarial Networks» [9], Krizhevsky A. et al. «ImageNet classification with deep convolutional neural networks» [10], Lin M. et al. «Network in network» [11], Zhu J.-Y. et al. «Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks» [12], He K. et al. «Deep residual learning for image recognition» [13], Szegedy C. et al. «Going Deeper with Convolutions» [14], Srivastava N. et al. «Dropout: a simple way to prevent neural networks from overfitting» [31], Ioffe S. et al. «Batch normalization: Accelerating deep network training by reducing internal covariate shift» [32], Szegedy C. et al. «Rethinking the Inception Architecture for Computer Vision» [35]. Также исследования крупных компаний в области машинного обучения рассмотрены на примере Google [4], Yandex [5], Facebook [6].

Помимо этого, пример практического применения многослойного персептрона рассмотрен в статье Кокшаровой Н. Б. «Использование многослойного персептрона для анализа вероятности банкротства компании» [20].

Для понимания механизмов функционирования нейросетей, а также градиентного и хеббовского обучения, помимо перечисленных учебных пособий, также в некоторой мере изучены работы, лежащие в основе этих алгоритмов. Для персептрона это McCulloch W. S., Pitts W «A logical Calculus of Ideas Immanent in Nervous Activity» [16], Rosenblatt F «Principles of neurodynamics: perceptrons and the theory of brain mechanisms» [19]; для

градиентного обучения Rumelhart D. E. et al. «Learning internal representations by error propagation» [28]; для хеббовского обучения Hebb D. O. «The organization of behavior: a neuropsychological theory» [30], Oja E. «A simplified neuron model as a principal component analyzer» [36], Kung S. Y., Diamantards K. I «A neural network leaming algorithm for adaptive principal component extraction (APEX)» [37].

Для понимания того, как можно организовать обучение с учителем в хеббовских нейросетях, рассматривалась работа Raphael H. L. et al. «Models of Acetylcholine and Dopamine Signals Differentially Improve Neural Representations» [38].

Рассмотрение сверточных нейронных сетей проходило на основе работ автора данной нейросетевой парадигмы: LeCun Ya., Bengio Yo «Convolutional Networks for Images, Speech, and Time Series» [21], LeCun Y. et al. «Backpropagation applied to handwritten zip code recognition» [22].

Такая нейросетевая парадигма как адаптивная резонансная теория, а также достижения в данной области были рассмотрены на примере статей Carpenter G., Grossberg S «A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine» [23], Rashchenko Y. V., Kozynchenko V. A. «New algorithms of an artificial neural network ART-1 training» [25], Rashchenko D. V. «Elimination of the search phase in the neural network ART-1 by changing the criterion of vectors similarity» [26] и диссертации Мищенко А. В. «Модель сознательного внимания и биоподобного анализа изображений на базе ансамбля АРТ-нейросетей» [24].

Следует отметить, что иногда для понимания деталей функционирования и реализации той или иной нейросетевой парадигмы удобно обращаться к электронным ресурсам, на которых материал преподносится популярно и доступно. К ним можно отнести «Хабрахабр» [34], [44], а также энциклопедию «Википедия» [17], [27].

Используемые обозначения

В данном разделе приведены некоторые соглашения по обозначению, принятые в рамках данной магистерской диссертации.

Векторы выделены жирным шрифтом, их элементы написаны обычным шрифтом с индексами, также все скалярные величины написаны обычным шрифтом. Например, вектор $\mathbf{y}$, его элемент y_i, вещественное число α.

Матрицы выделены заглавными буквами и жирным шрифтом, а их элементы написаны прописными буквами, обычным шрифтом и с двумя индексами (обозначающими строку и столбец матрицы соответственно). Например, матрица $\mathbf{W}$, ее элемент w_{ij}.

Верхний индекс в скобках используется для обозначения номера слоя. Например, $\mathbf{W}^{(k)}$ — весовая матрица k-го слоя.

Глава 1

Описание полносвязной и сверточной нейросетей

В данной главе рассматриваются базовые архитектуры многослойного персептрона и сверточной нейронной сети, принципы их функционирования на стадиях обучения (методом обратного распространения ошибки с градиентной оптимизацией) и предсказания.

1.1 Многослойный персептрон

Рассмотрим сперва однослойный персептрон, представленный на схеме рис. 1.1 [2]. Каждый элемент входа x_i связан с каждым нейроном единственного слоя персептрона посредством связи с весом w_{ij}, $i = \overline{1, M}$ — количество нейронов слоя, $j = \overline{1, N}$ — количество входных сигналов нейронной сети. Выход нейронного слоя вычисляется по следу-

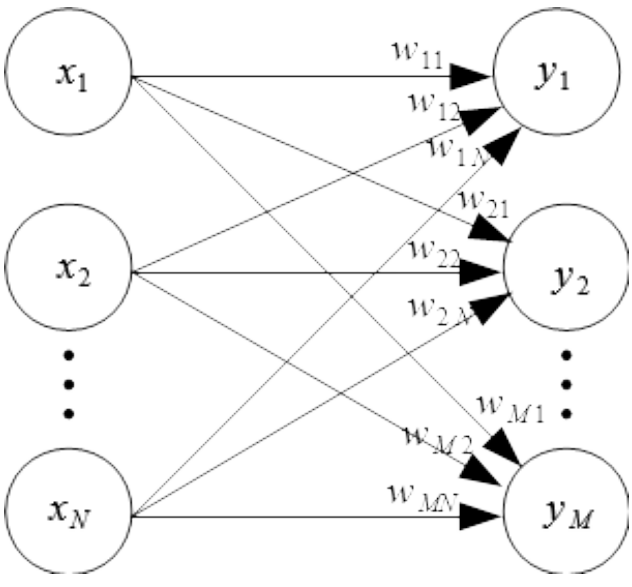

Рис. 1.1. Схема однослойного персептрона.

ющим формулам:

$$y_i = f(w_{ij}x_j), \tag{1.1}$$

либо в матричной форме:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x}), \tag{1.2}$$

где f — некоторая активационная (передаточная) функция, например, одна из функций, представленных в разделе «Введение».

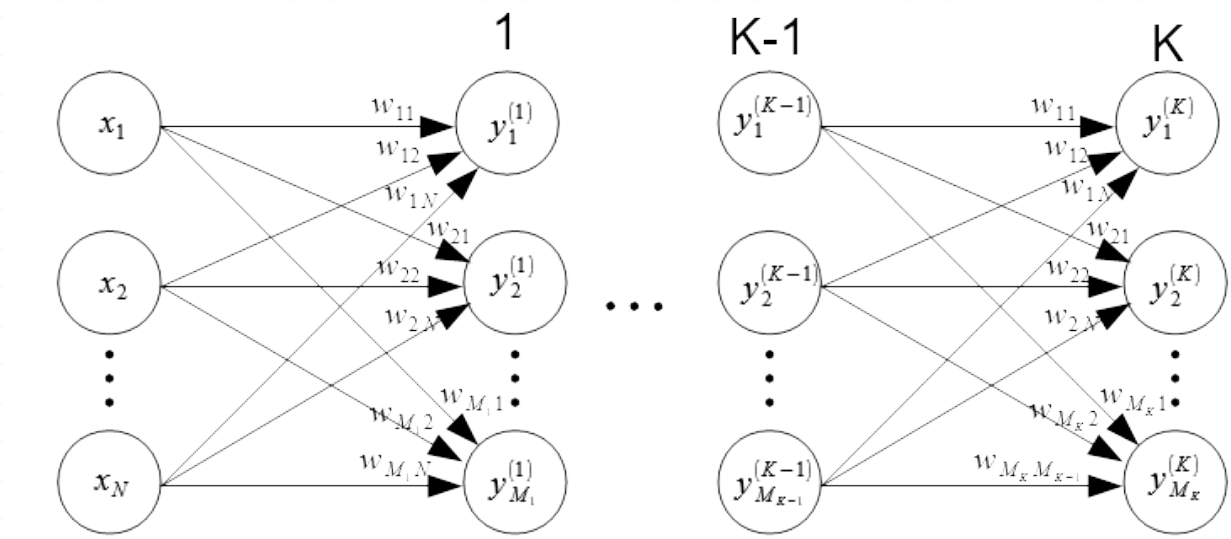

Рис. 1.2. Схема многослойного персептрона.

Схема полносвязной многослойной искусственной нейронной сети, или многослойного персептрона, представлена на рис. 1.2 [2]. В нем каждый нейрон первого слоя связан с каждым элементом входного вектора, и нейроны каждого последующего слоя связаны со всеми нейрона предыдущего слоя. Пусть $\mathbf{x} = \mathbf{y}^{(0)} \in \mathbb{R}^M$ — вход нейросети размера M; $\mathbf{y}^{(p)} \in \mathbb{R}^{M_p}$ — выход p-го слоя, состоящего из M_p нейронов, $p = \overline{1, K}$, где K — количество слоев; $\mathbf{W}^{(p)} \in \mathbb{R}^{M_p \times M_{p-1}}$ — матрица весов p-го слоя. Выход последнего слоя, в то же время являющийся и выходом персептрона, рассчитывается по итерационной формуле:

$$\mathbf{y}^{(K)} = f(\mathbf{W}^{(K)}\mathbf{y}^{(K-1)}) = f(\mathbf{W}^{(K)}f(\mathbf{W}^{(K-1)}\mathbf{y}^{(K-2)})) = \dots \tag{1.3}$$

1.2 Обучение многослойного персептрона методом градиентного спуска

Вернемся к однослойному персептрону, представленному в разделе 1.1. $\mathbf{x} \in \mathbb{R}^N$ — вход персептрона, $\mathbf{y} \in \mathbb{R}^M$ — выходной вектор. Пусть $\mathbf{d} \in \mathbb{R}^M$ — вектор ожидаемых выходных сигналов, тогда $(\mathbf{x}, \mathbf{d})$ — обучающая пара. Цель обучения состоит в подборе весовых коэффициентов w_{ij} таким образом, чтобы выход $\mathbf{y}$ нейросети совпадал с целевым выходом $\mathbf{d}$ с требуемой точностью. Таким образом, обучение заключается в решении задачи многомерной оптимизации в пространстве весов [33]:

$$\begin{aligned} F = \|\mathbf{d} - \mathbf{y}\| \to \min, \\ F = \sum_{i=0}^{M} \frac{1}{2}(d_i - y_i)^2 = \sum_{i=0}^{M} \frac{1}{2}(d_i - f(\sum_{j=0}^{N} w_{ij} x_j))^2. \end{aligned} \tag{1.4}$$

Для минимизации целевой функции часто используют метод градиентного спуска:

$$\begin{aligned} w_{ij}^* = w_{ij} - \alpha \frac{\partial F}{\partial w_{ij}}, \\ \frac{\partial F}{\partial w_{ij}} = -(d_i - f(\sum_{p=0}^{N} w_{ip} x_p)) f'(\sum_{p=0}^{N} w_{ip} x_p) x_j = \\ = -(d_i - y_i) f'(\sum_{p=0}^{N} w_{ip} x_p) x_j, \end{aligned} \tag{1.5}$$

где w_{ij}^* — весовые коэффициенты после одной итерации обучения, α — коэффициент обучения, который характеризует скорость сходимости алгоритма. Выбор альфа производится эмпирически из интервала $(0, 1)$. Подбор коэффициента обучения оказывает сильное влияние на эффективность обучения. Его величину можно задавать константой, но наиболее эффективным, но и трудоемким процессом является адаптивный подбор α на каждом шаге [33].

Для обучения многослойных нейронных сетей специально разработа-

на стратегия подбора весов, названная алгоритмом обратного распространения ошибки [33]. Пусть $\mathbf{d}$ — ожидаемый выход многослойного персептрона, $\mathbf{d} \in \mathbb{R}^{M_K}$, тогда $(\mathbf{y}^0, \mathbf{d})$ — обучающая пара. По аналогии с (1.4), функционал качества для многослойного персептрона запишется в виде:

$$F = \sum_{i=0}^{M_K} \frac{1}{2}(d_i - y_i^{(K)})^2, \tag{1.6}$$

а метод градиентного спуска:

$$w_{ij}^{(p)*} = w_{ij}^{(p)} - \alpha \frac{\partial F}{\partial w_{ij}^{(p)}}. \tag{1.7}$$

Рассмотрим последний, K-й слой и конкретизируем для него уравнение (1.6) через выход $(K-1)$-го слоя:

$$F = \sum_{i=0}^{M_K} \frac{1}{2}(d_i - f(\sum_{j=0}^{M_{K-1}} w_{ij}^{(K)} y_j^{(K-1)}))^2. \tag{1.8}$$

Возьмем частную производную по весам последнего слоя от уравнения (1.8):

$$\frac{\partial F}{\partial w_{ij}^{(K)}} = (d_i - y_i^{(K)}) f'(\sum_{p=0}^{M_{K-1}} w_{ip}^{(K)} y_p^{(K-1)}) y_j^{(K-1)}. \tag{1.9}$$

Обозначим

$$(d_i - y_i^{(K)}) f'(\sum_{p=0}^{M_{K-1}} w_{ip}^{(K)} y_p^{(K-1)}) = r_i^{(K)}, \tag{1.10}$$

тогда уравнение для производной перепишется в более простом виде:

$$\frac{\partial F}{\partial w_{ij}^{(K)}} = r_i^{(K)} y_j^{(K-1)}. \tag{1.11}$$

Найдем компоненты градиента для $(K-1)$-го слоя:

$$F = \sum_{i=0}^{M_K} \frac{1}{2}(d_i - f(\sum_{j=0}^{M_{K-1}} w_{ij}^{(K)} f(\sum_{q=0}^{M_{K-2}} w_{jq}^{(K-1)} y_q^{(K-2)})))^2, \tag{1.12}$$

$$\frac{\partial F}{\partial w_{jq}^{(K-1)}} =$$

$$= \sum_{i=0}^{M_K}(d_i - y_i^{(K)})f'(\sum_{m=0}^{M_{K-1}} w_{im}^{(K)} y_m^{(K-1)})w_{ij}^{(K)} f'(\sum_{n=0}^{M_{K-2}} w_{jn}^{(K-1)} y_n^{(K-2)})y_q^{(K-2)} =$$

$$= (\text{по (1.10)}) \sum_{i=0}^{M_K} r_i^{(K)} w_{ij}^{(K)} f'(\sum_{n=0}^{M_{K-2}} w_{jn}^{(K-1)} y_n^{(K-2)})y_q^{(K-2)} =$$

$$= r_j^{(K-1)} y_q^{(K-2)}, \quad (1.13)$$

где

$$r_j^{(K-1)} = \sum_{i=0}^{M_K} r_i^{(K)} w_{ij}^{(K)} f'(\sum_{n=0}^{M_{K-2}} w_{jn}^{(K-1)} y_n^{(K-2)}). \quad (1.14)$$

В случаях (1.11) и (1.13) описания градиента имеют одинаковую структуру и представляют произведение двух сигналов: входного сигнала нейрона и величины погрешности, переносимой на нейрон, от которого этот вход получен (с которым установлена взвешенная связь).

Аналогичные формулы будут получаться при продвижении от конца сети к ее началу. Для m-го слоя нейросети можно записать:

$$r_n^{(m)} = \sum_{i=0}^{M_{m+1}} r_i^{(m+1)} w_{in}^{(m+1)} f'(\sum_{j=0}^{M_{m-1}} w_{nj}^{(m)} y_j^{(m-1)}), \quad (1.15)$$

$$\frac{\partial F}{\partial w_{ij}^{(m)}} = r_i^{(m)} y_j^{(m-1)}, \quad (1.16)$$

где $r_n^{(m)}$ — ошибка, «распространяющаяся» на $(m-1)$-й слой, $r_i^{(m+1)}$ — ошибка, «пришедшая» с $(m+1)$-го слоя, $\frac{\partial F}{\partial w_{ij}^{(m)}}$ — частная производная функционала качества по весам m-го слоя; $m = \overline{1, K-1}$ — номер слоя, $i = \overline{0, M_{m+1}}$, $j = \overline{0, M_{m-1}}$ и $n = \overline{0, M_m}$ — итераторы по нейронам соответствующих слоев.

Таким образом, ошибка «распространяется» от последнего слоя сети к первому.

1.2.1 Алгоритм обучения

Ниже представлено краткое описание алгоритма обучения многослойного персептрона методом градиентной оптимизации с обратным распространением ошибки.

1. Инициализировать веса нейросети $\mathbf{W}^{(p)}, \quad p = \overline{1, K}, \quad K$ — количество нейронных слоев.

2. Задать максимально допустимое значение ε функционала качества F.

3. Подготовить обучающую пару $(\mathbf{x}, \mathbf{d})$, $\mathbf{x} \in \mathbb{R}^N, \mathbf{d} \in \mathbb{R}^{M_K}, \quad M_K$ — количество нейронов последнего слоя.

4. Произвести «прямое» распространение входных сигналов x_i и получить выходной вектор нейросети $\mathbf{y}^{(K)}$.

5. Вычислить значение функционала качества (1.6). Если $F \leqslant \varepsilon$, прекратить обучение.

6. Вычислить величину погрешности $\mathbf{r}^{(K)}$, переносимую на нейроны предпоследнего слоя, произведя матричное умножение с весовой матрицей $W^{(K)}$ для получения вектора, соразмерного с выходом $(K-1)$-го слоя: $\mathbf{r}^{(K)} = (\mathbf{d} - \mathbf{y}^{(K)}) f'(\mathbf{W}^{(K)}\mathbf{y}^{(K-1)})$, $\boldsymbol{\delta}^{(K-1)} = \mathbf{W}^{(K)T}\mathbf{r}^{(K)}$

7. Вычислить поправку к весам текущего слоя: $\Delta\mathbf{W}^{(K)} = \mathbf{r}^{(K)}\mathbf{y}^{(K-1)T}$ (T — знак транспонирования).

8. Вычислить величину погрешности, переносимую на нейроны $(K-2)$-го слоя: $\mathbf{r}^{(K-1)} = \boldsymbol{\delta}^{(K-1)} f'(\mathbf{W}^{(K-1)}\mathbf{y}^{(K-2)})$, $\boldsymbol{\delta}^{(K-2)} = \mathbf{W}^{(K-1)T}\mathbf{r}^{(K-1)}$

9. Вычислить поправку к весам текущего слоя: $\Delta\mathbf{W}^{(K-1)} = \mathbf{r}^{(K-1)}\mathbf{y}^{(K-2)T}$.

10. Повторить шаги 8–9 для всех слоев до второго включительно.

11. Для первого слоя вычислить поправку к весам: $\mathbf{r}^{(1)} = \mathbf{W}^{(2)T}\mathbf{r}^{(2)} f'(\mathbf{W}^{(1)}\mathbf{y}^{(0)})$, $\Delta\mathbf{W}^{(1)} = \mathbf{r}^{(1)}\mathbf{y}^{(0)T}$

12. Изменить веса нейросети в соответствии с формулой $\mathbf{W}^{(p)*} = \mathbf{W}^{(p)} - \alpha\Delta\mathbf{W}^{(p)}$.

13. Повторить шаги 3–12 до тех пор, пока ошибка не достигнет требуемого уровня (пункт 5).

1.3 Сверточная нейронная сеть

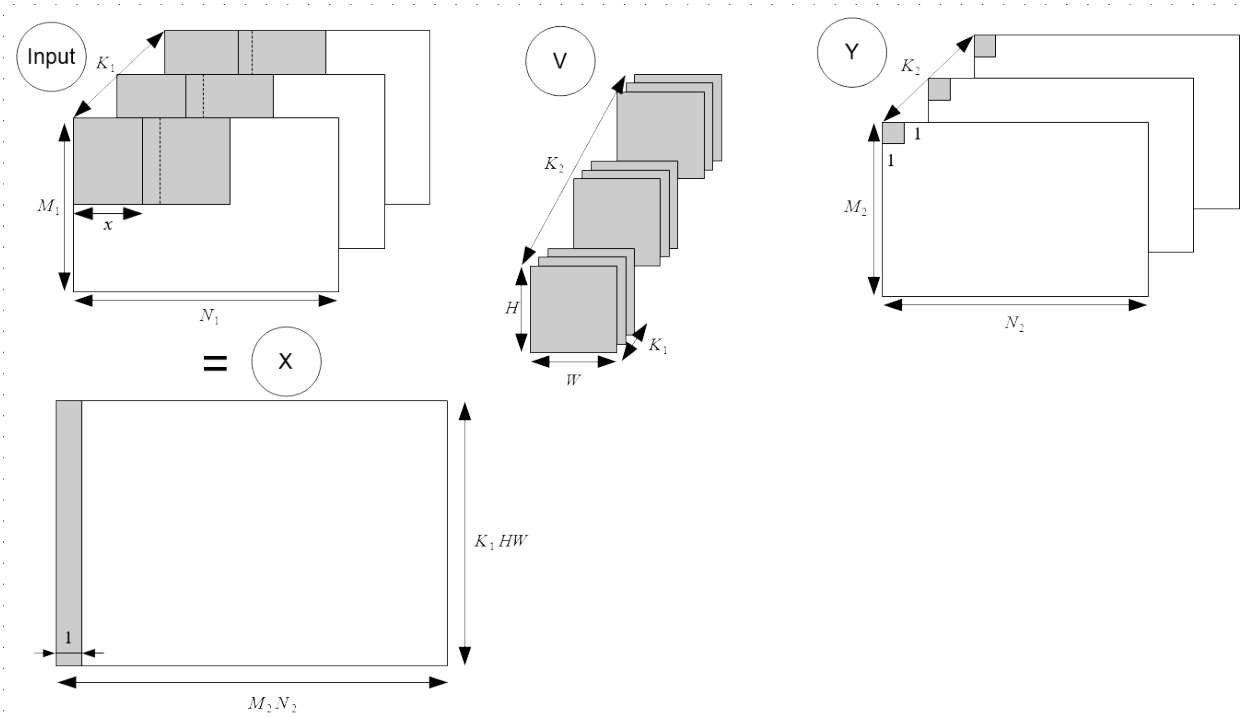

Рис. 1.3. Структура сверточного слоя. **Input** — вход, **V** — весовая матрица слоя, **Y** — выход; M_1, N_1 — высота и ширина входа; M_2, N_2 — высота и ширина выхода; H, W — высота и ширина ядра; K_1 — глубина входа и ядра; K_2 — глубина выхода и количество ядер; x — шаг смещения фильтра по горизонтали.

Вычисления в сверточном слое основаны на вычислениях полносвязных слоев, однако имеются существенные отличия [21]. Важной особенностью сверточных слоев является наличие разделенных весов — небольших весовых матриц, которые используются для вычислений всеми нейронами слоя (в отличие от полносвязного слоя, где у каждого нейрона свой собственный весовой коэффициент). Основу вычислений сверточного слоя составляют операции свертки ядер свертки слоя и некоторой области входного множества (рецептивного поля ядра), в результате которых получаются выходные карты признаков (выходы нейронов), являющиеся входами

следующего слоя.

На рис. 1.3 представлена схема сверточного слоя. На вход слой получает K_1-канальную карту признаков (K_1 плоскостей размера $M_1 \times N_1$). Так как сверточные нейронные сети применяются в основном в задачах машинного зрения, входом первого слоя обычно является 3-канальное (для цветных изображений) или 1-канальное (для изображений в градациях серого) изображение, соответственно признаками являются пиксели изображения. На последующих слоях в результате сверток выделяются высокоуровневые признаки, такие как линии разных направлений, углы и т.д.

Слой состоит из K_2 ядер свертки, или фильтров, каждый размером $H \times W \times K_1$. Каждое ядро ориентировано на выделение какого-то определенного признака. Каждый фильтр перемещается по входной карте признаков с шагом x по горизонтали и шагом y по вертикали, при этом перемещение сопровождается сверткой ядра с областью входной карты признаков, с которой совмещен фильтр. Величина, полученная в результате очередной свертки, заносится в соответствующую ячейку выходной карты признаков. Таким образом, области размера $M_1 \times N_1 \times K_1$ входной карты признаков сворачиваются в один элемент на p-ом листе выходной карты признаков, где $p = \overline{1, K_2}$ — номер ядра, с которым была произведена свертка, при этом обычно шаги перемещения фильтра меньше его размеров, что приводит к наложению рецептивных областей. Свертки играют роль детекторов признаков, поэтому глубина выходной карты признаков характеризует количество признаков, извлеченных на данном этапе вычислений, а высота и ширина — их пространственное взаимное расположение. К скалярному результату каждой свертки также могут применяться активационные функции, о которых шла речь в разделе «Введение».

Следующие формулы определяют высоту и ширину выхода сверточного слоя:

$$M_2 = \frac{M_1 - H}{y} + 1, \tag{1.17}$$

$$N_2 = \frac{N_1 - W}{x} + 1. \tag{1.18}$$

В сверточных нейронных сетях применяется также прием понижения размерности, известный как пулинг или субдискретизация (англ. pooling, subsampling). Метод эксплуатирует тот факт, что изображения обладают свойством локальной скоррелированности признаков: например, если говорить о пикселях, то соседние пиксели, как правило, не сильно отличаются друг от друга. Таким образом, если из нескольких соседних получить какой-либо агрегат, то потери информации будут незначительными [34]. Также прием пулинга добавляет в нейронную сеть некоторую инвариантность к масштабу входного изображения.

Слои пулинга очень похожи на сверточные слои, однако вместо операций свертки над рецептивными полями производятся более сложные нелинейные операции. Также шаг смещения ядра в таких слоях обычно равен размеру ядра. Наиболее распространены слои MaxPooling (рис. 1.4): в них области $H \times W \times K_1$ редуцируются в один элемент, являющийся максимальным в данной области. Также широко известна операция AveragePooling: в таких слоях области $H \times W \times K_1$ входной карты признаков превращаются в один элемент, являющийся средним арифметическим всех элементов рассматриваемой области.

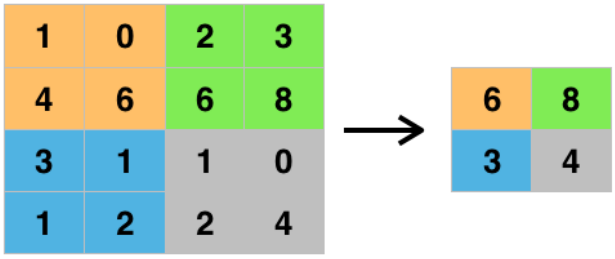

Рис. 1.4. Операция MaxPooling [27].

Стандартным шаблоном для построения сверточных сетей является чередование несколько раз слоев свертки и пулинга и затем добавление нескольких полносвязных слоев, которые в данной архитектуре играют роль дискриминатора. Пример такой сверточной нейронной сети представлен на рис. 5.

Как можно заметить, сверточные нейронные сети обладают довольно большим списком варьируемых параметров, а именно:

1. структура нейросети: количество слоев и их виды;

2. количество ядер в каждом слое;

3. размеры рецептивного поля;

4. шаги сдвига фильтра;

5. активационные функции слоев.

К этому списку также добавляются варьируемые параметры, пришедшие от полносвязных нейросетей: скорость обучения, критерий остановки обучения (приемлемая ошибка), структура полносвязных слоев. Такое количество гиперпараметров относят к недостаткам сверточных нейросетей, так как к тому же нет четких правил по их подбору. Однако большинство параметров интуитивно понятны и определимы исходя из требований решаемой задачи, а также существует большое количество различных эвристик, выработанных во время экспериментов с данным видом нейросетей [13, 35].

1.4 Градиентное обучение сверточной нейронной сети

Для удобства и наглядности формулировки алгоритма обучения трехмерный вход **Input** раскладывается в двумерную матрицу $\mathbf{X}$ размером $HWK_1 \times M_2N_2$ (используемые обозначения приведены на рис. 1.3). Для создания первого столбца берется левый верхний прямоугольный сегмент входной карты признаков размером $H \times W$ и записывается построчно в вектор. Затем берется такой же сегмент второго листа и записывается в конец получившегося на предыдущем шаге вектора. Такая процедура производится со всеми K_1 листами входа, в результате чего получается первый столбец матрицы $\mathbf{X}$. Для получения второго столбца происходит смещение на каждом листе входа на x элементов вправо, и процедура повторяется. Когда первая «строка» прямоугольных сегментов заканчивается, происходит переход на следующую «строку» посредством смещения вниз на y элементов, где y — шаг смещения фильтра по вертикали. При этом если при очередном смещении оставшаяся часть строки (столбца) меньше по длине W (H), то эта оставшаяся часть игнорируется. Следует отметить,

что при $x < W$ или $y < H$ некоторые элементы входа будут повторяться в матрице $\mathbf{X}$.

Используя введенные обозначения, можно описать вычисления в сверточном слое при прямом распространении следующим образом:

$$Y_{k_2 j} = \sum_{i=0}^{K_1 HW} V_{k_2 i} X_{ij}, \tag{1.19}$$

где $i = \overline{0, K_1 HW}$, $i = k_1 HW + hW + w$, $j = \overline{0, M_2 N_2}$, $j = n_2 M_2 + m_2$, $k_1 = \overline{0, K_1}$, $k_2 = \overline{0, K_2}$, $m_2 = \overline{0, M_2}$, $n_2 = \overline{0, N_2}$, $h = \overline{0, H}$, $w = \overline{0, W}$.

При распространении сигнала со сверточной части на персептрон происходит взвешенное связывание каждого нейрона сверточного слоя (элементы выходной карты признаков) с каждым нейроном полносвязного слоя. Это можно реализовать преобразованием трехмерной матрицы выхода размера $K_2 \times M_2 \times N_2$ в вектор размера $K_2 M_2 N_2 \times 1$, который поступает на вход полносвязного слоя. Таким образом, весовая матрица полносвязного слоя имеет размер $M \times K_2 M_2 N_2$, где M — количество нейронов слоя персептрона.

Алгоритм обратного распространения ошибки с градиентной оптимизацией для сверточного слоя формулируется на основе аналогичного алгоритма для полносвязного слоя [22]. Рассмотрим некоторый сверточный слой p. Со слоя $p+1$ принимается сигнал погрешности $\delta^{(p)}_{K_2 M_2 N_2}$. Далее вычисляется поправка к весам текущего слоя:

$$r^{(p)}_{k_2 ij} = \delta^{(p)}_{k_2 j} f'(V^{(p)}_{k_2 i} X^{(p)}_{ij}), \tag{1.20}$$

$$\Delta V^{(p)}_{k_2 i} = \sum_{j=0}^{M_2 N_2} r^{(p)}_{k_2 ij} X^{(p)}_{ij}, \tag{1.21}$$

и погрешность, переносимая на $(p-1)$-й слой:

$$\delta^{(p-1)}_{ij} = \sum_{k_2=0}^{K_2} r^{(p)}_{k_2 ij} V^{(p)}_{k_2 i}. \tag{1.22}$$

При этом $\boldsymbol{\delta}^{(p-1)}$ является двумерной матрицей размера $K_1HW \times M_2N_2$, аналогичной матрице $\mathbf{X}$. Для передачи ее в $(p-1)$-й слой необходимо преобразовать ее к виду трехмерной матрицы размера $K_1 \times M_2 \times N_2$, для этого нужно повторить преобразование $\mathbf{Input} \to \mathbf{X}$ в обратном порядке, с учетом шагов смещения рецептивного поля. При наложении рецептивных полей производиться суммирование элементов, попавших в пересечение.

При обратном распространении ошибки между полносвязной и сверточной частью нейросети в сверточный слой приходит вектор погрешности размера $K_2M_2N_2 \times 1$, который необходимо преобразовать в трехмерную матрицу $K_2 \times M_2 \times N_2$ для дальнейшего использования в сверточном слое.

Также следует отметить про обратное распространение ошибки в слоях пулинга. Пусть слой пулинга имеет порядковый номер p. При использовании MaxPooling или AveragePooling ядра этих слоев не используют обучаемых весов, поэтому задача состоит только в преобразовании матрицы погрешности размера $K_2 \times M_2 \times N_2$, полученную от $(k+1)$-го слоя, в матрицу погрешности размера $K_1 \times M_1 \times N_1$ для передачи ее в $(k-1)$-й слой. Так как операция пулинга заключается в редукции рецептивной области входа в один элемент выхода, при обратном распространении необходимо выполнить обратную операцию, «расширив» один элемент в K_1HW элементов. Поскольку эта операция не является однозначной, используются различные методы для «развертки», одним из которых является следующий:

$$\delta_{i,j}^{(p-1)} = \sum_{k_2=0}^{K_2} \frac{\delta_{k_2 j}^{(p)}}{HWK_1}. \tag{1.23}$$

Матрица $\boldsymbol{\delta}^{(p-1)}$ здесь также имеет двумерный вид размера $K_1HW \times M_2N_2$, и ее необходимо преобразовать в матрицу размера $K_1 \times M_1 \times N_1$. При этом, как было отмечено, шаги смещения ядра пулинга равны ширине и высоте рецептивного поля, поэтому при преобразовании не возникает наложения областей.

1.4.1 Алгоритм обучения

Схема обучения сверточной нейронной сети аналогична алгоритму обучения многослойного персептрона. Вначале необходимо инициализировать веса всех слоев нейросети, задать максимально допустимую ошибку, подготовить обучающее множество. Затем производится прямое распространение изображения из первой обучающей пары через все слои нейросети, полученный выход нейросети сравнивается с требуемым выходом, и вычисляется ошибка. Если ошибка меньше максимально допустимого значения, то обучение прекращается, иначе происходит обратное распространение ошибки от последнего слоя к первому и вычисление в каждом слое поправки к весам. После этого производится модификация весов нейросети в соответствии с правилом градиентного спуска, и все итерации повторяются для следующей обучающей пары.

Глава 2

Обучение по правилу Хебба

В данной главе производится обзор правил изменения весовых коэффициентов на основе постулата обучения Хебба и предлагаются некоторые модифицированные варианты. При этом рассматриваются однонейронные и многонейронные сети. Также предлагается несколько вариантов обучения хеббовских нейронных сетей при наличии размеченных данных (с учителем) с опорой на существующие работы в данном направлении. В завершающем разделе главы хеббоподобные правила обучения формулируются для сверточных слоев.

2.1 Фильтр Хебба

Фильтр Хебба (см. рис. 2.1) — это линейный нейрон с хеббовским правилом настройки синаптических весов. Постулат обучения Хебба, сформулированный нейрофизиологом Дональдом Хеббом, звучит следующим образом [30]:

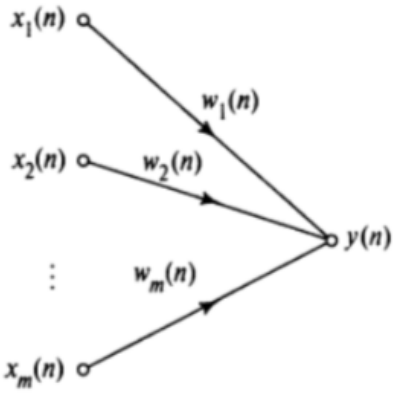

Рис. 2.1. Фильтр Хебба [29].

«Если аксон клетки A находится на достаточно близком расстоянии от клетки B и постоянно или периодически участвует в ее возбуждении, наблюдается процесс

метаболических изменений в одном или обоих нейронах, выражающийся в том, что эффективность нейрона A как одного из возбудителей нейрона B возрастает».

Правило Хебба, адаптированное для использования в искусственных нейронных сетях, выглядит следующим образом [29]:

$$w_i(n+1) = w_i(n) + \alpha y(n) x_i(n), \quad i = \overline{1, m}, \tag{2.1}$$

где w_i — синаптический вес, n — дискретный шаг по времени, m — число элементов входного вектора $\mathbf{x}$, α — параметр интенсивности (скорость) обучения, x_i — компонента входного вектора, y — выход нейрона. При этом выход нейрона определяется как:

$$y = \sum_{i=1}^{m} w_i x_i. \tag{2.2}$$

Правило (2.1) приводит к неограниченному росту весов, поэтому часто используется нормализация. Нормализованное правило Хебба в приближении при малом α называется правилом Ойя [36] и выглядит следующим образом:

$$w_i(n+1) = w_i(n) + \alpha y(n)(x_i(n) - y(n) w_i(n)). \tag{2.3}$$

Далее в данной работе для обозначения данного правила будет использовано название Oja_1.

Фильтр Хебба обладает свойством выделения первой главной компоненты (направления наибольшей дисперсии) распределения входных векторов. Для самоорганизующегося нейрона, обучающегося по алгоритму (2.3), доказана сходимость его вектора весов к вектору первой главной компоненты при некоторых условиях [29]. Наиболее важным из них для практического применения алгоритма является ограничение на параметр обучения: значения $\vec{\alpha}(n)$ должны быть убывающей последовательностью поло-

жительных действительных чисел, такой, что:

$$\sum_{n=1}^{\infty} \alpha(n) = \infty, \tag{2.4a}$$

$$\sum_{n=1}^{\infty} \alpha^p(n) < \infty, \quad p > 1, \tag{2.4b}$$

$$\alpha(n) \to 0 \text{ при } n \to \infty. \tag{2.4c}$$

При практическом использовании в нейронных сетях выбор α, строго удовлетворяющего данному условию, уменьшит пластичность модели, поэтому параметру α часто назначают некоторое малое положительное значение, руководствуясь требованиями области приложения [29].

2.1.1 Модель со слоем нейронов.

Фильтр Хебба — однонейронная модель, извлекающая первую главную компоненту из входного сигнала. Она может быть расширена до сети прямого распространения с одним слоем линейных нейронов (рис. 2.2) для анализа нескольких главных компонент [29]. Веса слоя нейронов изменяются в соответствии с обобщенной формой правила обучения Ойя (2.5), которое называется обобщенным алгоритмом обучения Хебба. Далее в данной работе будет применяться обозначение «Oja_n_theory».

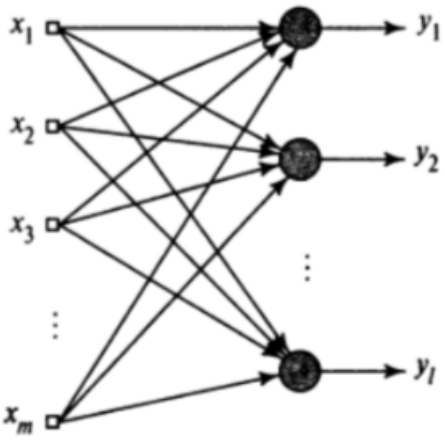

Рис. 2.2. Сеть прямого распространения с одним слоем линейных нейронов [29].

$$w_{ji}(n+1) = w_{ji}(n) + \alpha y_j(n)(x_i(n) - \sum_{k=1}^{j} w_{ki}(n) y_k(n)) \tag{2.5}$$
$$i = \overline{1,m}, \quad j = \overline{1,l},$$

где l — количество нейронов в слое, а выход нейрона j определяется по следующей формуле:

$$y_j(n) = \sum_{i=1}^{m} w_{ji}(n)x_i(n). \tag{2.6}$$

Для данного алгоритма также доказана сходимость весов нейронов к первым l главным компонентам входных данных (при дополнительных условиях) [29]. Следует отметить, что при обучении по алгоритму Oja_n_theory нейроны слоя упорядочены по убыванию главных компонент.

2.1.2 Модель с латеральным торможением.

Алгоритм адаптивного извлечения главных компонент [37] (англ. adaptive principal components extraction — APEX, далее — APEX_theory) использует, помимо прямых, еще и обратные связи (рис. 2.3). Выход нейрона вычисляется по следующей формуле:

$$y_j(n) = \sum_{i=1}^{m} w_{ji}(n)x_i(n) + \sum_{i=1}^{j-1} a_{ji}(n)y_i(n), \tag{2.7}$$

где a_{ji} — латеральный вес. Из формулы видно, что алгоритм является итеративным: выход j-го нейрона вычисляется на основе $j-1$ предыдущих нейронов.

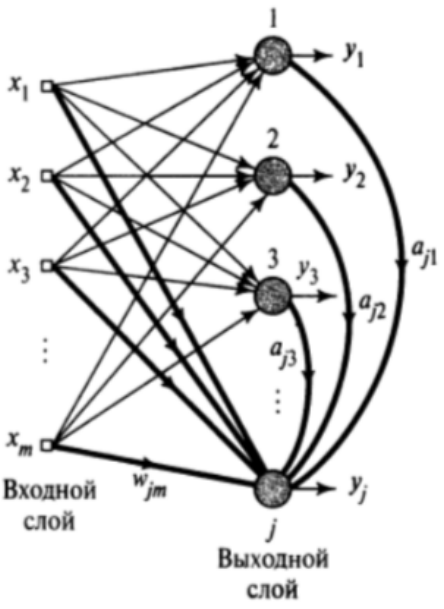

Рис. 2.3. Фильтр Хебба [29].

Правило обучения весов определяется как:

$$w_{ji}(n+1) = w_{ji}(n) + \alpha y_j(n)(x_i(n) - y_j(n)w_{ji}(n)), \tag{2.8}$$

$$a_{jk}(n+1) = a_{jk}(n) - \alpha y_j(n)(y_k(n) + y_j(n)a_{jk}(n)). \tag{2.9}$$

Алгоритм APEX_theory без обратных связей схож с алгоритмом

Oja_1. Однако если рассмотреть слой нейронов, обучающихся по правилу Oja_1, то их веса будут сходиться к одному и тому же вектору (первой главной компоненте). Введение латеральных связей обеспечивает конкуренцию нейронов слоя, что приводит к их декорреляции, и в результате веса нейронов сходятся к разным векторам главных компонент.

Алгоритм Oja_n_theory также предусматривает конкуренцию нейронов слоя, что заставляет выделять их разные главные компоненты, но это достигается за счет использования при обучении нейрона информации о весах других нейронов слоя. Преимуществом APEX_theory перед Oja_n_theory является то, что достигается тот же эффект, но только за счет локальной информации, что более сходно с нейробиологическими структурами и удовлетворяет основным принципам, на которых изначально основано обучение Хебба [29].

Для алгоритма APEX_theory также доказана сходимость весов $\mathbf{W}$ нейронов к первым l главным компонентам входа [37]. При этом, аналогично Oja_n_theory, нейроны слоя упорядочены по убыванию главных компонентов.

2.1.3 Симметричные алгоритмы

В Oja_n_theory и APEX_theory наблюдается неравнозначность нейронов слоя, что не соответствует устройству реальных биологических структур, поэтому в случае с Oja_n_theory также рассматривается немного видоизмененный симметричный алгоритм [29] (здесь и далее — с условным обозначением «Oja_n»). Симметричность достигается за счет использования при обучении j-го нейрона информации о весах всех l нейронов данного слоя (по сравнению с Oja_n_theory):

$$w_{ji}(n+1) = w_{ji}(n) + \alpha y_j(n)(x_i(n) - \sum_{k=1}^{l} w_{ki}(n) y_k(n)). \tag{2.10}$$

В данной работе предлагается по аналогии с Oja_n симметричная версия алгоритма APEX_theory (далее — APEX). В алгоритме APEX j-

й нейрон, в отличие от APEX_theory, имеет латеральные связи со всеми нейронами слоя, поэтому его выход вычисляется таким образом:

$$y_j(n) = \sum_{i=1}^{m} w_{ji}(n)x_i(n) + \sum_{i=1}^{l} a_{ji}(n)y_i(n), \quad a_{jj} = 1. \tag{2.11}$$

Правила изменения весов в алгоритме APEX идентичны (2.8, 2.9).

2.1.4 Применение активационных функций в хеббоподобных нейронах

Обычно в фильтрах Хебба используются линейные функции активации, и именно такие случаи рассматривались до этого в данной главе. Также предлагается рассматривать фильтры Хебба с нелинейными функциями активации. Тогда формулы (2.6) и (2.11) соответственно преобразуются в:

$$y_j(n) = f(\sum_{i=1}^{m} w_{ij}(n)x_i(n)), \tag{2.12}$$

$$y_j(n) = f(\sum_{i=1}^{m} w_{ji}(n)x_i(n) + \sum_{i=1}^{l} a_{ji}(n)y_i(n)), \quad a_{jj} = 1, \tag{2.13}$$

где f — это некоторая нелинейная функция, например, из набора, приведенного в разделе «Введение».

2.2 Обучение с учителем в нейросетях с фильтрами Хебба

Для реализации обучения с учителем в нейросетях с хеббоподобными нейронами используются различные методы [38]. В данной работе рассмотрено несколько конфигураций. Для обучения с учителем в задаче распознавания имеется некоторый целевой вектор $\mathbf{y}_{target}$ размера l, который состоит из нулей и одной единицы. Компонента, равная единице, опре-

деляет принадлежность входного образа конкретному классу. При этом для получения выхода нейросети, который удобно сравнить с целевым вектором, в нейронах последнего слоя применяется активационная функция Softmax (5) или Hardmax (6).

DirectionLast. В данном методе сравниваются выходной вектор нейросети и целевой. Если максимальный элемент выхода соответствует максимальному элементу целевого вектора (по порядковому номеру), то обучение последнего слоя проводится по уже известным формулам (2.5), (2.8), (2.9), (2.10) при $\alpha > 0$. В противном случае в указанных формулах коэффициент обучения берется $\alpha < 0$ (антихеббовское обучение). Обучение всех остальных слоев всегда проводится при $\alpha > 0$.

DirectionAll. Данный метод аналогичен DirectionLast, но в случае несоответствия выходного вектора нейросети целевому вектору обучение с параметром $\alpha < 0$ проводится во всех слоях нейросети.

TargetedLast. Обучение происходит по правилам (2.5), (2.8), (2.9), (2.10) с положительным параметром обучения, но на последнем слое при обучении вместо выходов нейронов используются соответствующие элементы целевого вектора.

GradientLastDirectionAll. Последний слой нейросетей, обучающихся по данному алгоритму, полносвязный, и его обучение проводится методом градиентной оптимизации (см. главу 1). Для остальных слоев обучение проводится согласно алгоритму DirectionAll.

GradientLast. Последний слой (полносвязный) обучается аналогично GradientLastDirectionAll, все остальные слои обучаются по формулам (2.5), (2.8), (2.9), (2.10) с положительным α.

2.3 Применение правила Хебба для обучения сверточных нейронных сетей

Традиционным подходом для обучения сверточных нейронных сетей является метод градиентного спуска с обратным распространением ошибки (см. главу 1, раздел 1.4). В данной работе предлагается использовать алгоритмы, основанные на правиле Хебба, для обучения сверточных нейронных сетей.

Алгоритмы обучения Хебба (2.1), Oja_1 (2.3) и Oja_n (2.10) адаптированы для сверточного слоя следующим образом:

- правило Хебба:

$$V_{k_2k_1hw}(n+1) = V_{k_2k_1hw}(n) + \alpha \frac{\sum\limits_{m_2,n_2=0}^{M_2-1,N_2-1} X_{\beta\gamma}(n) Y_{k_2m_2n_2}(n)}{M_2N_2}; \qquad (2.14)$$

- правило Oja_1:

$$V_{k_2k_1hw}(n+1) = V_{k_2k_1hw}(n) + \\ + \alpha \frac{\sum\limits_{m_2,n_2=0}^{M_2-1,N_2-1} Y_{k_2m_2n_2}(n)(X_{\beta\gamma}(n) - V_{k_2k_1hw}(n) Y_{k_2m_2n_2}(n))}{M_2N_2}; \quad (2.15)$$

- правило Oja_n:

$$V_{k_2k_1hw}(n+1) = V_{k_2k_1hw}(n) + \\ + \alpha \frac{\sum\limits_{m_2,n_2=0}^{M_2-1,N_2-1} Y_{k_2m_2n_2}(n)(X_{\beta\gamma}(n) - \sum\limits_{a=0}^{K_2-1} Y_{am_2n_2}(n) V_{ak_1hw}(n))}{M_2N_2}. \quad (2.16)$$

В формулах использованы следующие обозначения: $\beta = k_1HW + hW +$

$w, \quad \gamma = m_2 N_2 + n_2, \quad k_1 = \overline{0, K_1 - 1}, \quad k_2 = \overline{0, K_2 - 1}, \quad h = \overline{1, H}, \quad w = \overline{1, W}$. Используемые матрицы $\mathbf{X}, \mathbf{Y}, \mathbf{V}$ аналогичны тем, что были введены в главе 1, разделе 1.3.

Следует отметить, что при вычислении поправки к весу для учета выходов всех нейронов слоя (их количество $K_2 \times M_2 \times N_2$) используется среднее арифметическое.

Глава 3

Программная реализация, тестирование и анализ результатов

В данной главе приведены некоторые детали реализации описанных в работе алгоритмов, а также подробное описание проведенных экспериментов и анализ результатов.

3.1 Особенности реализации

Рассмотренные в работе нейронные сети (полносвязная нейросеть, сверточная нейросеть) и различные алгоритмы обучения (градиентное обучение, хеббовское обучение) реализованы в виде компьютерной модели на языке C++, проведено сравнительное тестирование различных моделей с использованием базы изображений MNIST [39] на примере следующих задач машинного зрения: выделение главных компонент во входном множестве изображений и классификация.

Подготовленный программный комплекс создан с соблюдением принципов методологии объектно-ориентированного программирования. Для обеспечения более эффективных вычислений стадии функционирования, вклю-

чающие матричное умножение, реализованы с помощью CBLAS [40].

Из соображений учебной тренировки все алгоритмы были реализованы «с нуля», без использования дополнительных специализированных библиотек машинного обучения. Однако следует отметить, что в условиях решения реальных бизнес-задач для реализации базовых алгоритмов, лежащих в основе функционирования ИНС, более целесообразно использовать готовые библиотеки машинного обучения, например [41], которые удобны в использовании, хорошо проверены и эффективно используют доступные вычислительные мощности (CPU и GPU). Таким образом, можно сосредоточиться на более глубоких исследованиях в рамках поставленной задачи для улучшения качества программного продукта.

Эксперименты проводились на персональном компьютере со следующими характеристиками: процессор Intel(R) Core(TM) i3-3220 CPU 3.30GHz*4, доступная память ОЗУ 8 Гб. К сожалению, из-за недостатка вычислительных мощностей и неориентированности программы на вычисления на GPU эксперименты на более глубоких нейросетях, чем те, о которых будет говориться далее, оказались очень времязатратными, из-за чего их не удалось провести, поэтому такие эксперименты в работе не упоминаются.

3.2 Эксперименты по выделению главных компонент

В данном разделе рассматривается задача выделения главных компонент. Производится сравнение результатов работы различных хеббоподобных однослойных моделей и статистического метода главных компонент.

Тестирование проводилось в течение $N = 10^5$ итераций предъявления случайного образа из обучающей выборки. Во время экспериментов исследовались следующие аспекты:

1. устойчивость алгоритмов при различных постоянных значениях параметра обучения α: неустойчивость констатировалась в случае сильного

увеличения весов за проведенное время обучения. Начальное значение α для всех моделей было выбрано 0.1. Если для некоторой конфигурации выявлялась неустойчивость при текущем α, то значение параметра обучения для этой нейросети уменьшалось в 10 раз, и обучение начиналось заново. Такое определение неустойчивости и метод настройки α использован во всех последующих экспериментах;

2. сравнение несимметричных алгоритмов APEX_theory и Oja_n_theory с их симметричными аналогами APEX и Oja_n соответственно;

3. применение нейронов с активационными функциями.

В таблице 3.1 приведены конфигурации нейросетей, для которых было проведено тестирование в рамках данного исследования. Всего было проанализировано 16 моделей с различными сочетаниями правила обучения и активационной функции. В последнем столбце таблицы указано значение скорости обучения для каждой конфигурации, которое удалось подобрать опытным путем в рамках эксперимента.

На рис. 3.1 приведена визуализация результатов работы алгоритмов по выделению первой главной компоненты входного множества: рисунки 1–16 соответствуют алгоритмам 1–16 в таблице 3.1, рисунок 17 — методу главных компонент (principal component analysis, PCA) [42]. Для выделения главных компонент методом PCA была реализована вспомогательная программа на языке Python с использованием библиотеки машинного обучения Scikit-Learn [43].

Анализ результатов тестирования, представленных на рис. 3.1, позволяет сделать вывод, что безусловно хеббоподобные алгоритмы справляются с задачей выделения главных компонент. При этом лучший результат, судя по изображению, показали конфигурации 9 и 13: APEX+Linear и APEX_theory+Linear. Выделенная ими первая главная компонента наиболее схожа с результатом работы PCA. При этом следует отметить, что веса нейросетей, которые при обучении могли оказаться как положительными, так и отрицательными, проецировались на промежуток $[-1, 1]$, и при визуализации весовых матриц в виде изображений в градациях серого уровень

Таблица 3.1. Эксперимент 1. Тестирование однослойных нейросетей.

№	Правило обучения	Активационная функция	α
1	Oja_n	Linear	0.01
2	Oja_n	ReLU	0.01
3	Oja_n	Sigmoid	0.1
4	Oja_n	Sigmoid_Antisimmetric	0.1
5	Oja_n_theory	Linear	0.01
6	Oja_n_theory	ReLU	0.01
7	Oja_n_theory	Sigmoid	0.1
8	Oja_n_theory	Sigmoid_Antisimmetric	0.1
9	APEX	Linear	0.001
10	APEX	ReLU	0.01
11	APEX	Sigmoid	0.1
12	APEX	Sigmoid_Antisimmetric	0.1
13	APEX_theory	Linear	0.001
14	APEX_theory	ReLU	0.01
15	APEX_theory	Sigmoid	0.1
16	APEX_theory	Sigmoid_Antisimmetric	0.1

яркости пикселя, соответствующего конкретному весу, масштабировался в промежутке $[0, 255]$. Таким образом, векторы первой главной компоненты у моделей 9 и 13 коллинеарны, но противоположно направлены. То есть можно считать, что эти модели выделили одну и ту же первую главную компоненту входного набора данных.

Всего в рамках данного эксперимента каждым алгоритмом было выделено 10 первых главных компонент. Анализ таблицы 3.1, результатов для первой главной компоненты на рис. 3.1, а также для других компонент, позволил сделать следующие выводы:

1. устойчивость алгоритмов (по крайней мере в течение времени проведения данного тестирования) с активационными функциями Linear и ReLU наблюдается при меньших скоростях обучения, чем с Sigmoid и Sigmoid_Antisimmetric;
2. симметричные алгоритмы APEX и Oja_n сравнимы с несимметричными APEX_theory и Oja_n_theory (в смысле выделения главных компонент);

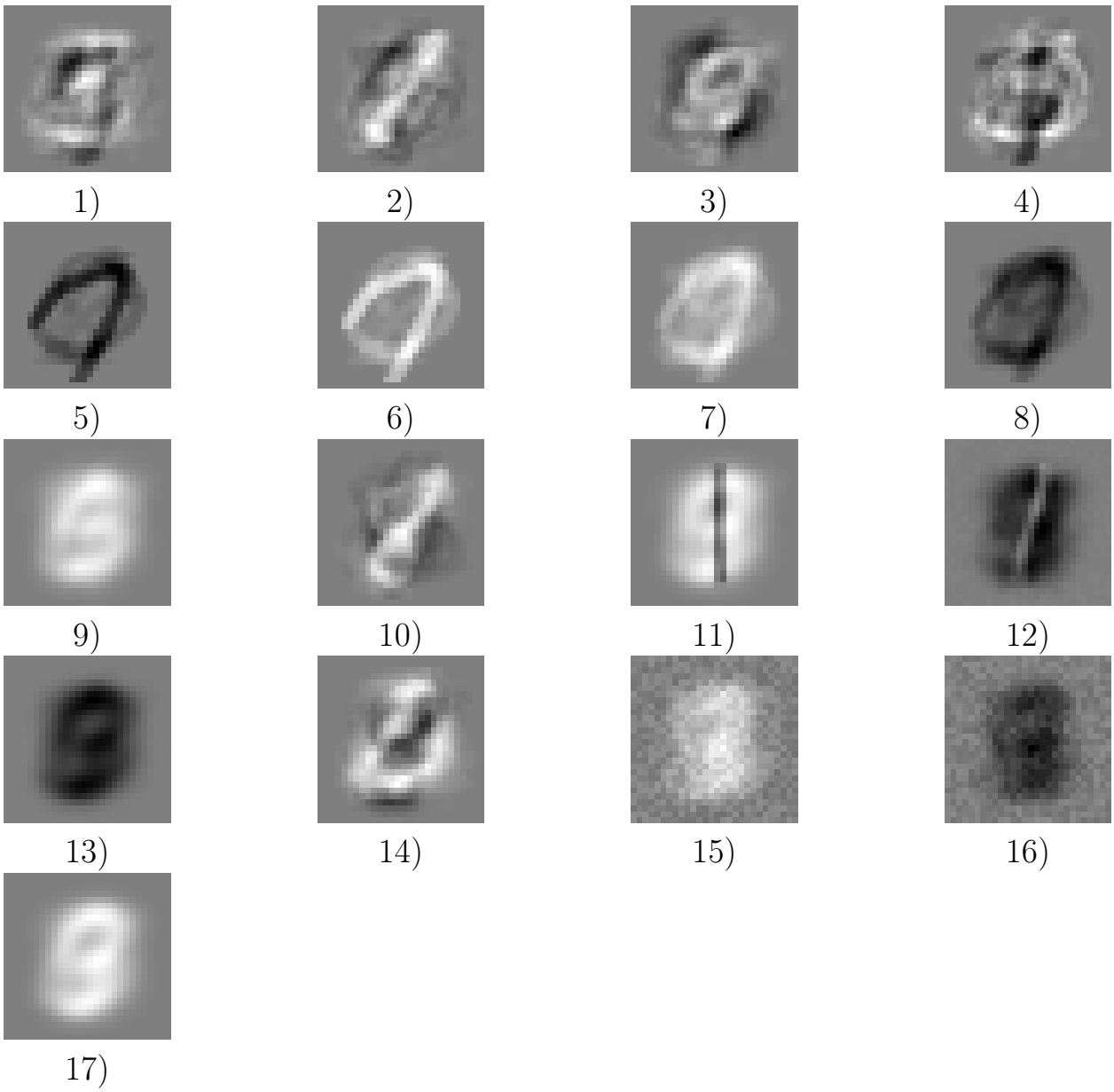

Рис. 3.1. Первая главная компонента, выделенная различными алгоритмами: 1–16 соответствуют алгоритмам 1–16 из таблицы 3.1, 17 — PCA.

3. алгоритмы без применения нелинейной активационной функции (с функцией Linear) показали себя значительно лучше алгоритмов с нелинейными активационными функциями.

3.3 Обучение с учителем неглубоких полносвязных нейросетей

В данном разделе описаны эксперименты по обучению с учителем неглубоких (3 слоя) нейросетевых моделей с фильтрами Хебба. Здесь и далее работа нейросетей оценивается на примере задачи классификации.

Во всех слоях, кроме последнего, использовались линейные нейроны (по опыту последнего эксперимента), в последнем слое применена активационная функция Softmax (5) или Hardmax (6). Вначале проведено предварительное непродолжительное тестирование ($3 \cdot 10^4$ итераций), во время которого были выделены наиболее успешные методы обучения с учителем для дальнейшей работы. Во второй части эксперимента проведено окончательное тестирование продолжительностью $5 \cdot 10^4$ итераций для выбранных моделей.

3.3.1 Предварительное тестирование

На данном этапе во всех тестируемых конфигурациях в качестве правила изменения весов использовалось Oja_n_theory (2.5). Всего было проверено 12 конфигураций. Результаты эксперимента представлены в таблице 3.2. Столбец α содержит значения параметров обучения, при которых соответствующая нейронная сеть в течение времени проведения тестирования оставалась устойчивой. Последний столбец таблицы содержит ошибки соответствующих моделей, полученные при тестировании и усредненные по количеству проведенных итераций тестирования (образов в тестовой выборке). Обучающая выборка не включает в себя тестовые образы. Ошибка (здесь и далее) рассматривается в промежутке $[0, 2]$.

В результате данного эксперимента для дальнейшего анализа были выбраны конфигурации 3 и 8 из таблицы 3.2, которые использовали метод обучения с учителем GradientLast. Они показали лучшие результаты по ошибке и по выбранному параметру обучения. Для остальных нейросетей наибольший оптимальный коэффициент α оказался слишком мал. Можно

Таблица 3.2. Эксперимент 2, часть 1.

№	Активационная функция последнего слоя	Метод обучения с учителем	α	Средняя ошибка
1	Softmax	DirectionAll	0.000001	1,839
2	Softmax	DirectionLast	0.000001	1,798
3	Softmax	GradientLast	0.001000	1,110
4	Softmax	GradientLast DirectionAll	0.000001	1,809
5	Softmax	TargetedLast	0.000001	1,764
6	Hardmax	DirectionAll	0.000001	1,653
7	Hardmax	DirectionLast	0.000001	1,811
8	Hardmax	GradientLast	0.001000	0,694
9	Hardmax	GradientLast DirectionAll	0.000001	1,720
10	Hardmax	TargetedLast	0.000001	1,835

увидеть это на графике изменения ошибки (рис. 3.2), не вводя строгого ограничения на допустимую нижнюю границу α. На графике видно, что

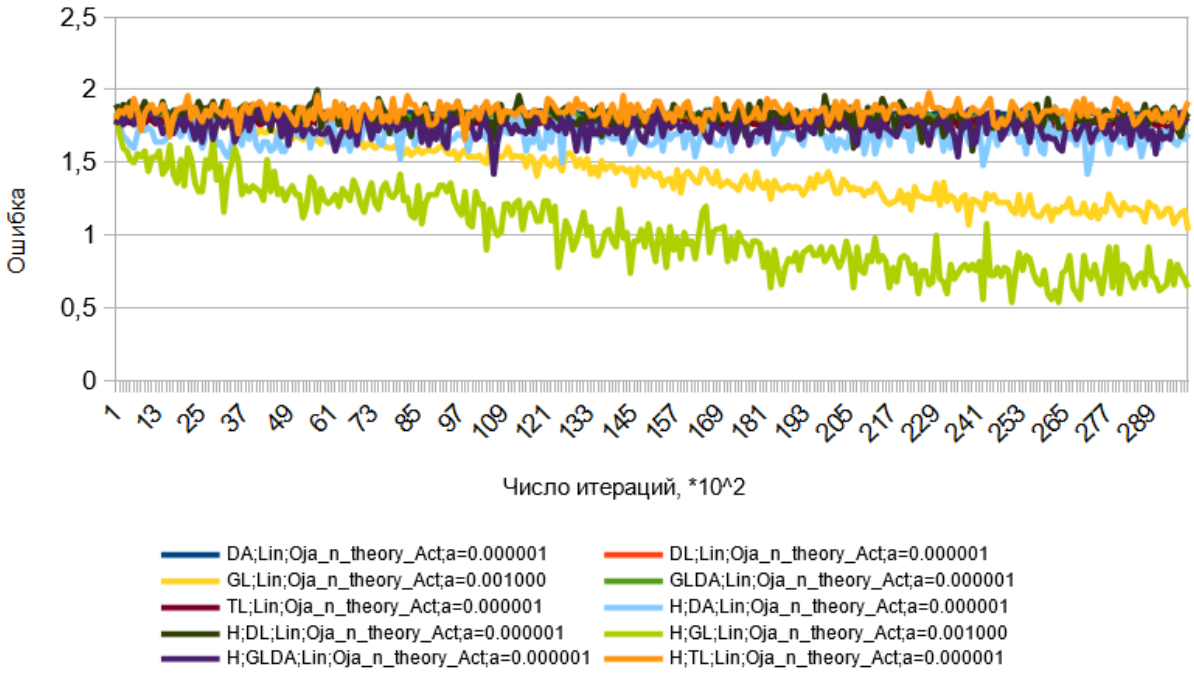

Рис. 3.2. Графики изменения ошибки во время обучения для нейросетей в эксперименте 2, часть 1.

только у моделей с GradientLast (в легенде графика он отмечен как «GL»)

ошибка убывала, а у остальных колебалась около некоторого высокого значения в течение всего обучения.

3.3.2 Окончательное тестирование

В данном эксперименте рассматриваются многослойные полносвязные нейронные сети с фиксированным правилом обучения с учителем, но различными хеббоподобными правилами изменения весов. Результаты тестирования этих нейросетей также сравниваются с результатами нейросети с градиентным обучением.

Проведено сравнительное тестирование 45 конфигураций нейронных сетей (табл. 3.3): 44 нейросети с хеббоподобными правилами модификации весов, которые обучались с учителем согласно методу GradientLast, и 1 нейросеть с обучением методом обратного распространения ошибки с использованием градиентной оптимизации. Варьировались такие параметры, как активационная функция последнего слоя (Softmax или Hardmax), активационная функция скрытых слоев (Linear, ReLU, Sigmoid, Sigmoid_Antisimmetric) и правило обучения (Oja_n, Oja_n_theory, Oja_n_Act, Oja_n_theory_Act, APEX_Act, APEX_theory_Act). При этом активационная функция в скрытых слоях применялась в двух вариантах: до обучения (т. е. в обучении участвовал уже активированный выход) и после обучения (в обучении участвовал неактивированный выход). В столбце «Правило обучения» таблицы 3.3 это обозначается наличием или отсутствием постфикса «_Act». Естественно, для активационной функции «Linear» рассматривался только один из вариантов, так как второй дает те же результаты. Также из-за особенностей реализации среди APEX-подобных алгоритмов рассматривались только модели с применением активационной функции до обучения (APEX_Act и APEX_theory_Act).

Для каждой модели приведены полученные в ходе эксперимента коэффициенты α и ошибки. Нейросети в таблице отсортированы по возрастанию ошибки.

Таблица 3.3. Эксперимент 2, часть 2.

№	Акт. функция последнего слоя	Акт. функция	Правило обучения	α	Средняя ошибка
1	Softmax	ReLU	Grad	0.100000	0.116
2	Hardmax	Linear	Oja_n	0.010000	0.292
3	Hardmax	ReLU	Oja_n_Act	0.010000	0.298
4	Hardmax	Sigmoid_ Antisimmetric	Oja_n	0.010000	0.301
5	Hardmax	ReLU	Oja_n	0.010000	0.318
6	Hardmax	Sigmoid_ Antisimmetric	Oja_n_Act	0.010000	0.377
7	Softmax	Linear	Oja_n	0.010000	0.378
8	Hardmax	Linear	APEX_theory_ Act	0.001000	0.494
9	Hardmax	ReLU	APEX_theory_ Act	0.001000	0.558
10	Hardmax	Linear	Oja_n_theory	0.001000	0.597
11	Hardmax	Linear	APEX_Act	0.001000	0.612
12	Hardmax	ReLU	APEX_Act	0.001000	0.649
13	Hardmax	Sigmoid	Oja_n_Act	0.010000	0.734
14	Softmax	ReLU	Oja_n	0.010000	0.768
15	Softmax	ReLU	Oja_n_Act	0.010000	0.796
16	Softmax	ReLU	APEX_theory_ Act	0.001000	0.833
17	Softmax	Linear	APEX_Act	0.001000	0.891
18	Softmax	Linear	Oja_n_theory	0.001000	0.893
19	Softmax	Sigmoid_ Antisimmetric	Oja_n	0.010000	0.971
20	Softmax	Linear	APEX_theory_ Act	0.001000	0.986

21	Softmax	Sigmoid_ Antisimmetric	Oja_n_Act	0.010000	1.006
22	Hardmax	Sigmoid	APEX_Act	0.010000	1.155
23	Hardmax	ReLU	Oja_n_theory	0.001000	1.177
24	Softmax	ReLU	APEX_Act	0.001000	1.207
25	Softmax	Sigmoid	Oja_n_Act	0.010000	1.430
26	Hardmax	ReLU	Oja_n_theory_ Act	0.010000	1.476
27	Hardmax	Sigmoid_ Antisimmetric	Oja_n_theory	0.010000	1.523
28	Hardmax	Sigmoid_ Antisimmetric	APEX_Act	0.001000	1.590
29	Hardmax	Sigmoid	Oja_n	0.010000	1.654
30	Hardmax	Sigmoid	Oja_n_theory	0.001000	1.680
31	Softmax	ReLU	Oja_n_theory	0.001000	1.710
32	Softmax	ReLU	Oja_n_theory_ Act	0.010000	1.723
33	Softmax	Sigmoid	APEX_Act	0.010000	1.781
34	Softmax	Sigmoid_ Antisimmetric	Oja_n_theory	0.010000	1.781
35	Softmax	Sigmoid_ Antisimmetric	APEX_Act	0.001000	1.784
36	Hardmax	Sigmoid	Oja_n_theory_ Act	0.010000	1.788
37	Softmax	Sigmoid	Oja_n_theory	0.001000	1.796
38	Softmax	Sigmoid	Oja_n	0.010000	1.797
39	Softmax	Sigmoid	APEX_theory_ Act	0.100000	1.799
40	Softmax	Sigmoid_ Antisimmetric	APEX_theory_ Act	0.100000	1.799
41	Hardmax	Sigmoid_ Antisimmetric	Oja_n_theory_ Act	0.100000	1.801

42	Softmax	Sigmoid	Oja_n_theory_Act	0.100000	1.803
43	Hardmax	Sigmoid_Antisimmetric	APEX_theory_Act	0.100000	1.804
44	Hardmax	Sigmoid	APEX_theory_Act	0.100000	1.810
45	Softmax	Sigmoid_Antisimmetric	Oja_n_theory_Act	0.100000	1.832

В качестве наиболее оптимальных конфигураций были выбраны первые 10 нейросетей в таблице 3.3. Для них на рис. 3.3 приведены графики изменения ошибки обучения.

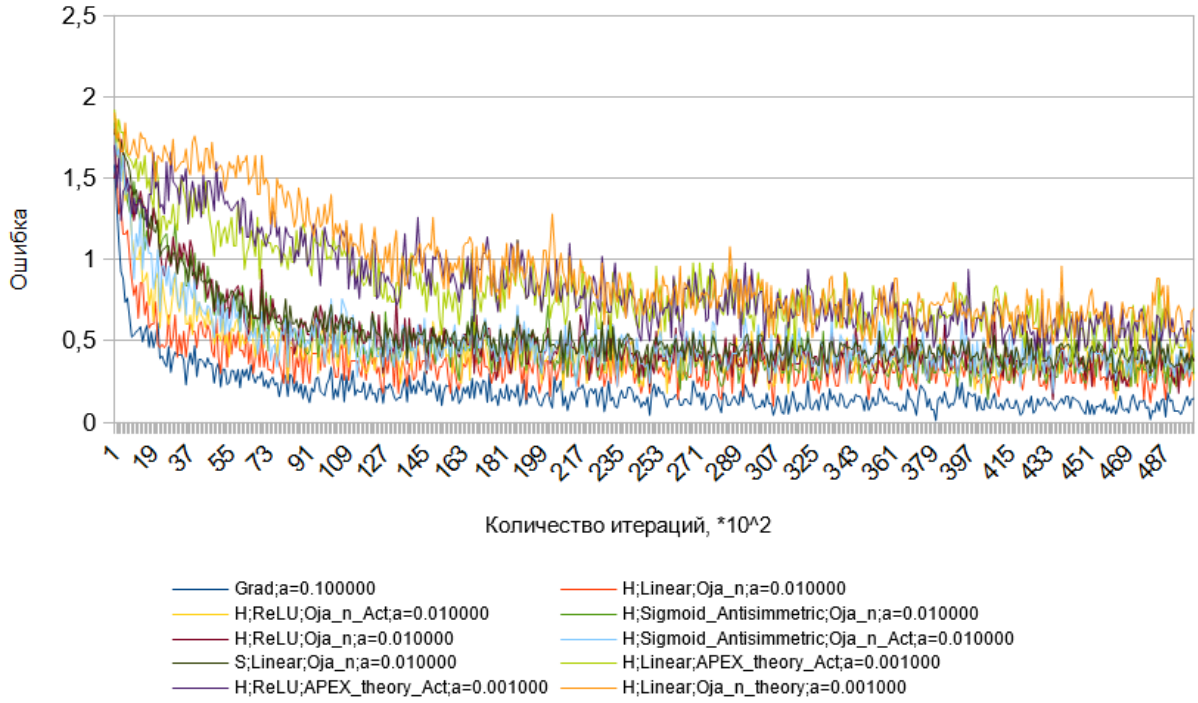

Рис. 3.3. Графики изменения ошибки для десяти лучших нейросетей в эксперименте 2, часть 2.

3.4 Обучение сверточной нейронной сети

3.4.1 Эксперимент с неглубокой архитектурой

На данном этапе шесть лучших конфигураций «активационная функция — правило обучения» из таблицы 3.3 были использованы для обучения неглубокой сверточной нейронной. Рассматривалась сверточная сеть со следующей архитектурой:

1. первый слой — сверточный:
   - количество ядер: 10;
   - высота ядра: 5;
   - ширина ядра: 5;
   - шаг смещения фильтра по горизонтали: 3;
   - шаг смещения фильтра по вертикали: 3;
   - количество нейронов (вычисляется из размера входа и гиперпараметров): 640;
2. второй слой — полносвязный, 10 нейронов

Было проведено $5 \cdot 10^4$ итераций. Результаты эксперимента приведены в таблице 3.4. Всего было протестировано 12 нейросетей: 6 дополнительных моделей было получено заменой хеббоподобного правила обучения (модели 1–6 в таблице) методом обратного распространения ошибки с градиентной оптимизацией (модели 7–12). В таблице для каждой нейросети указаны подобранное значение скорости обучения и средняя тестовая ошибка.

По результатам данного эксперимента лучшими сверточными сетями среди «хеббовских» оказались нейросети 1 и 6, т. е. с линейной активационной функцией и правилом обучения Oja_n. При этом они не сильно уступают по качеству своим «градиентным» аналогам.

Таблица 3.4

№	Акт. функция последнего слоя	Акт. функция скрытых слоев	Правило обучения	α	Ошибка
1	Hardmax	Lin	Oja_n	0.01	0,3456
2	Hardmax	ReLU	Oja_n_Act	0.10	1,1170
3	Hardmax	Sigmoid _Antisimmetric	Oja_n	0.10	0,8194
4	Hardmax	ReLU	Oja_n	0.10	1,5534
5	Hardmax	Sigmoid _Antisimmetric	Oja_n_Act	0.10	0,6844
6	Softmax	Lin	Oja_n	0.01	0,3270
7	Hardmax	Lin	Grad(CNN)	0.10	0,4350
8	Hardmax	ReLU	Grad(CNN)	0.10	1,8040
9	Hardmax	Sigmoid _Antisimmetric	Grad(CNN)	0.10	0,4330
10	Hardmax	ReLU	Grad(CNN)	0.10	1,8040
11	Hardmax	Sigmoid _Antisimmetric	Grad(CNN)	0.10	0,4332
12	Softmax	Lin	Grad(CNN)	0.01	0,2825

Таблица 3.5

№	Акт. функция последнего слоя	Акт. функция скрытых слоев	Метод обучения	α	Ошибка
1	Hardmax	Lin	Oja_n	0.000010	0,3866
2	Softmax	Lin	Oja_n	0.000010	0,391305
3	Softmax	Lin	Grad	0.001000	0,244029

3.4.2 Эксперимент с глубокой архитектурой

На данном этапе рассматривается наилучшее сочетание «правило обучения — активационная функция» по итогам эксперимента с неглубокой архитектурой (линейная активационная функция скрытых слоев и прави-

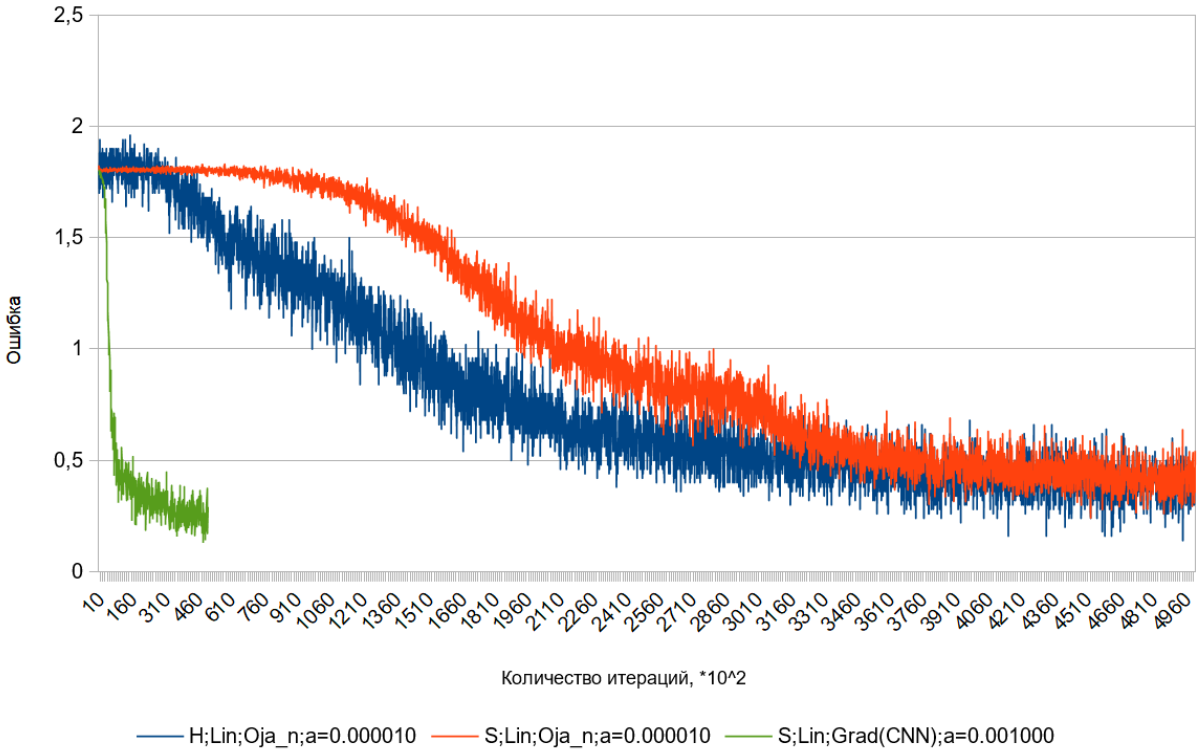

Рис. 3.4. Графики изменения ошибки для глубоких сверточных нейросетей.

ло обучения Oja_n) при хеббовском обучении более глубокой сверточной нейронной сети, результаты сравниваются с градиентным обучением аналогичной по архитектуре нейросети. При этом архитектура выбрана с учетом рекомендаций статьи [44]:

1. сверточный слой, количество ядер 32;
2. сверточный слой, аналогичный первому;
3. слой maxpooling;
4. сверточный слой, количество ядер 64;
5. сверточный слой, аналогичный четвертому слою;
6. слой maxpooling, аналогичный второму слою;
7. полносвязный слой, 512 нейронов;
8. полносвязный выходной слой, 10 нейронов.

Размер ядер сверточных слоев равен 3, слоев субдискретизации — 4. Обучение проводилось в течение $5 \cdot 10^5$ итераций.

Обучение градиентной нейросети проводилось в течение $5 \cdot 10^4$ итераций со скоростью обучения $\alpha = 10^{-3}$.

В таблице 3.5 представлены полученные значения скорости обучения α и ошибки на тестовом множестве для исследуемых моделей. На рис. 3.4 приведены графики изменения ошибки во время обучения. Таким образом, можно заключить, что в результате данного эксперимента удалось достичь качества хеббовского обучения, сравнимого с результатами градиентного обучения (при одинаковой конфигурации нейросетей), при том что хеббовская нейросеть является самоорганизующейся системой, и только последний слой обучается с учителем. Более продолжительное время хеббовского обучения, возможно, связано со значительно меньшей его скоростью обучения по сравнению с градиентным.

Также следует отметить, что полученный уровень ошибки, очевидно, не претендует на лидерство в задаче классификации рукописных цифр, даже классификация значительно более сложных датасетов производится сейчас с более высоким качеством с помощью новейших нейросетевых парадигм (например, [13]). Но в рамках данного эксперимента не стояло задачи получить модель, опережающую по качеству существующие методы, целью было провести сравнительное обучение градиентного и хеббовского обучения, поэтому архитектуры нейросетей были выбраны достаточно простыми, и во время обучения не применялось особых приемов, зарекомендовавших себя в машинном обучении, с помощью которых, вероятно, удалось бы получить более высокое качество [32, 31].

Выводы

В рамках данной магистерской диссертации рассмотрены базовые архитектуры полносвязной многослойной нейронной сети (многослойного персептрона) и сверточной нейронной сети, а также их принципы функционирования в режимах обучения и предсказания. Рассмотрены алгоритмы обучения двух видов: наиболее популярный метод градиентного спуска с обратным распространением ошибки и менее распространенные алгоритмы обучения, основанные на правиле Хебба. Главным преимуществом хеббовских методов является высокая локальность и отсутствие необходимости вычислять градиент в многомерном пространстве весовых коэффициентов, что изначально ведет к невозможности появления такой проблемы алгоритма обратного распространения ошибки как затухающие градиенты, а также отсутствию требования дифференцируемости активационной функции нейронов.

Существуют различные правила модификации весов, основанные на базовом правиле Хебба. В данной работе рассмотрены следующие правила:

1. базовое правило Хебба (для одного нейрона);
2. правило Ойя (нормализованное правило Хебба);
3. обобщенное правило Хебба (обобщенная форма правила Ойя, для слоя нейронов);
4. симметричное обобщенное правило Хебба;
5. APEX — алгоритм адаптивного извлечения главных компонент (фильтр Хебба для слоя нейронов с латеральными связями).

Для последнего алгоритма предложена модификация, заключающаяся в добавлении симметричности связей по аналогии с симметричным обобщенным правилом Хебба.

Хеббовские нейросети в основном позиционируются как самоорганизующиеся модели, однако в работе предложено несколько алгоритмических

конфигураций для обучения многослойных фильтров Хебба на размеченной выборке, то есть с учителем. В основном принцип этих алгоритмов заключается в том, что выход нейросети сравнивается с целевым значением, и в зависимости от их расхождения нейроны сети обучаются по хеббовскому или антихеббовскому правилу. При этом локальность обучения не нарушается. Также рассматривается синергия градиентного и хеббовского обучения, когда нейроны последнего слоя нейросети обучаются в соответствии с методом градиентного спуска, а нейроны остальных слоев — по хеббовскому (или антихеббовскому) алгоритму.

Для обучения сверточных нейронных сетей повсеместно применяется градиентная оптимизация с обратным распространением ошибки. В данной научно-исследовательской работе предложен алгоритм модификации весов сверточных нейросетей согласно принципам хеббовского обучения.

Рассмотренные в работе искусственные нейронные сети и алгоритмы обучения реализованы в виде компьютерной программы, и проведено обширное сравнительное тестирование градиентного и хеббовских алгоритмов обучения, проанализированы его результаты. В случае хеббовских нейросетей были исследованы различные конфигурации обучения, а именно:

1. различные хеббоподобные правила обучения (перечислены выше в данном разделе);
2. применение нейронов с активационными функциями (изначально хеббовские правила обучения формулируются для линейных нейронов);
3. различные значения скорости обучения;
4. различные подходы к обучению с учителем;

При этом на протяжении экспериментов были рассмотрены различные архитектуры нейросетей: от однослойных до более глубоких (8 слоев).

Фильтры Хебба известны как успешные модели для извлечения главных компонент. В данной работе были проведены эксперименты по обучению нейронных сетей извлечению главных компонент входного множе-

ства векторов, а также классификации образов. Все эксперименты проводились с использованием свободно распространяемой базы рукописных цифр MNIST.

В случае первой задачи сравнение работы хеббовских алгоритмов проводилось с результатами статистического метода главных компонент на том же входном множестве. Большинство тестируемых хеббовских алгоритмов успешно справились с задачей, но наиболее эффективными оказались правила APEX с линейными нейронами.

В задаче классификации образов использовались приемы обучения с учителем, и наиболее успешным по результатам многочисленных экспериментов оказался подход, при котором последний нейронный слой обучался в соответствии с градиентной оптимизацией, а другие слои — по симметричному обобщенному правилу Хебба, при этом нейроны — линейные (кроме нейронов последнего слоя, к выходам которых применена функция Softmax или Hardmax). Завершающим экспериментом стало обучение глубоких (8 слоев) сверточных нейронных сетей с градиентным и хеббовским обучением (в соответствии с описанным выше успешным подходом). В результате этого эксперимента удалось достичь качества хеббовского обучения сравнимого с качеством градиентного обучения.

Стоит отметить, что исследования по хеббовскому обучения слабо представлены в литературе. Проведенные в рамках данной работы исследования показали, что хеббовское обучение жизнеспособно, применимо в прикладных задачах машинного зрения, а также может использоваться для различных современных нейросетевых архитектур (многослойный персептрон и сверточные нейронные сети). Биоподобные принципы самоорганизации, заложенные в основы хеббовского обучения, вызывают интерес, и исследования в данной области кажутся перспективными.

Заключение

В данной магистерской диссертации изучены свойства и проанализированы некоторые проблемы фильтра Хебба и его различных реализаций,

а также предложены для них некоторые модификации. Также предложено несколько вариантов реализации хеббовского обучения в задаче обучения с учителем. Хеббовсие алгоритмы обучения адаптированы для модификации весов сверточной нейронной сети. Проведены эксперименты по обучению хеббовских нейросетей выделению главных компонент и классификации образов, при этом результаты обучения сравнивались соответственно с работой статистического метода главных компонент и нейросетей, обучающихся методом градиентной оптимизации. Архитектуры нейросетей варьировались от однослойного персептрона до многослойной сверточной, и в результате экспериментов удалось достичь достаточно высокого качества хеббовского обучения нейросетей, включая глубокую сверточную сеть. Это позволяет сделать вывод, что обучение по Хеббу применимо в сверточных нейросетях, являющихся на данный момент мощнейшим инструментом в области машинного зрения, что открывает широкие возможности. Свойство локальности хеббовских алгоритмов делает обучение потенциально более быстрым, чем градиентное, так как нет необходимости распространять сигналы и в прямом, и в обратном направлениях, и изначально избавляет обучение от таких трудностей градиентной оптимизации, как, например, проблема затухающих градиентов. Благодаря этому становится возможным обучение глубоких нейросетей с использованием правил Хебба.

Литература

[1] Gatys L. A., Ecker A. S., Bethge M. Texture synthesis using convolutional neural networks // Proceedings of the 28th International Conference on Neural Information Processing Systems — Volume 1. Montreal, Canada: MIT Press Cambridge, 2015. P. 262–270.

[2] Уоссермен Ф. Нейрокомпьютерная техника. М.: Мир, 1992. 184 с.

[3] Параллельные вычисления CUDA // NVIDIA URL: http://www.nvidia.ru/object/cuda-parallel-computing-ru.html (дата обращения: 23.04.2018).

[4] Research at Google URL: https://research.google.com/ (дата обращения: 23.04.2018).

[5] Research at Yandex https://research.yandex.com/ (дата обращения: 23.04.2018).

[6] Facebook AI Research (FAIR) // Facebook research https://research.fb.com/category/facebook-ai-research-fair/ (дата обращения: 23.04.2018).

[7] Luan F., Paris S., Shechtman E., Bala E. Deep photo style transfer // IEEE Conference on Computer Vision and Pattern Recognition. 2017. P. 6997–7005.

[8] Zheng S., Jayasumana S., Romera-Paredes B., Vineet V., Su Z., Du D., Huang C., Torr P. H. S., Conditional Random Fields as Recurrent Neural Networks // Proceedings of the 2015 IEEE International Conference on Computer Vision. 2015. P. 1529–1537.

[9] Isola P., Zhu J. Y., Zhou T., Efros A. A. Image-to-Image Translation with Conditional Adversarial Networks // IEEE Conference on Computer Vision and Pattern Recognition. 2017. P. 5967–5976

[10] Krizhevsky A., Sutskever Il., Hinton G. E. ImageNet classification with deep convolutional neural networks // Proceedings of the 25th International Conference on Neural Information Processing Systems. 2012. P. 1097–1105.

[11] Lin, M., Chen, Q., Yan, S. Network in network // Proceedings of the International Conference on Learning Representations. 2014.

[12] Zhu J.-Y., Park T., Isola P., Efros A. A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks // The IEEE International Conference on Computer Vision. 2017. P. 2242–2251.

[13] He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition // IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 770–778.

[14] Szegedy C., Liu W., Jia Ya., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. Going Deeper with Convolutions // IEEE Conference on Computer Vision and Pattern Recognition. 2015. P. 1–9.

[15] Дубынин В. А. Регуляторные системы организма человека. М.: Дрофа, 2003. 368 с.

[16] McCulloch W. S., Pitts W. A logical Calculus of Ideas Immanent in Nervous Activity // The bulletin of mathematical biophysics. 1943. P. 115–133.

[17] Перцептрон // Википедия URL: https://ru.wikipedia.org/wiki/Перцептрон (дата обращения: 23.04.2018).

[18] Redozubov A. Holographic memory: a novel model of information processing by neuronal microcircuits // The Physics of the Mind and Brain Disorders. Cham: Springer, 2017. P. 271–295.

[19] Rosenblatt F. Principles of neurodynamics: perceptrons and the theory of brain mechanisms. Washington: Spartan books, 1962. 616 p.

[20] Кокшарова Н. Б. Использование многослойного персептрона для анализа вероятности банкротства компании // Экономика и предпринимательство. 2013. №2-2 (41-2). С. 747–750.

[21] LeCun Ya., Bengio Yo. Convolutional Networks for Images, Speech, and Time Series // Michael A. Arbib, The Handbook of Brain Theory and Neural Networks. London: Massachusetts Institute of Technology, 2003. P. 276–278.

[22] LeCun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W., Jackel L. D. Backpropagation applied to handwritten zip code recognition // Neural Computation. 1989. No 1. P. 541–551.

[23] Carpenter G., Grossberg S A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine // Computer vision, graphics, and image processing. 1987. № 37. С. 54–115.

[24] Мищенко А. В. Модель сознательного внимания и биоподобного анализа изображений на базе ансамбля АРТ-нейросетей: дис. ... канд. физ.-мат. наук: 05.13.18. СПб, 2010. 226 с.

[25] Rashchenko Y. V., Kozynchenko V. A. New algorithms of an artificial neural network ART-1 training // IEEE 2015 International Conference «Stability and Control Processes» in memory of V. I. Zubov (SCP). 2015. P. 663–664.

[26] Rashchenko D. V. Elimination of the search phase in the neural network ART-1 by changing the criterion of vectors similarity // IEEE 2015 International Conference «Stability and Control Processes» in memory of V. I. Zubov (SCP). 2015. P. 661–662.

[27] Сверточная нейронная сеть // Википедия URL: https://ru.wikipedia.org/wiki/Сверточная_нейронная_сеть (дата обращения: 23.04.2018).

[28] Rumelhart D. E., Hinton G. E., Williams R. J. Learning internal representations by error propagation // Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. 1986. P. 318–362.

[29] Хайкин С. Нейронные сети: полный курс. 2 изд. М.: ООО «И. Д. Вильямс», 2006. 1104 с.

[30] Hebb D. O. The organization of behavior: a neuropsychological theory, New York: Wiley, 1949.

[31] Srivastava N., Hinton G., Krizhevsky Al., Sutskever Il., Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting // The Journal of Machine Learning Research. 2014. №15-1. P. 1929–1958.

[32] Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift // Proceedings of The 32nd International Conference on Machine Learning. 2015. P. 448-Ц456.

[33] Осовский С. Нейронные сети для обработки информации. М.: Финансы и статистика, 2002. 344 с.

[34] Обзор топологий глубоких сверточных нейронных сетей // Хабрахабр URL: https://m.habrahabr.ru/company/mailru/blog/311706/ (дата обращения: 23.04.2018).

[35] Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z. Rethinking the Inception Architecture for Computer Vision // IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 2818-2826.

[36] Oja E. A simplified neuron model as a principal component analyzer // J. Math. Biology. 1982. No 15. P. 267–273.

[37] Kung S. Y., Diamantards K. I A neural network leaming algorithm for adaptive principal component extraction (APEX) // Proceedings of International Conference on Acoustics, Speech, and Signal Processing. 1990. P. 861–864.

[38] Raphael H. L., Jorg L., Klaus O. Models of Acetylcholine and Dopamine Signals Differentially Improve Neural Representations // Frontiers in Computational Neuroscience. 2017. P. 54–72.

[39] The MNIST database of handwritten digits // Yann LeCun URL: http://yann.lecun.com/exdb/mnist/ (дата обращения: 23.04.2018).

[40] BLAS // Netlib Repository URL: http://www.netlib.org/blas/index.html (дата обращения: 23.04.2018).

[41] Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., Kudlur M., Levenberg J., Monga R., Moore S., Murray D. G., Steiner B., Tucker P., Vasudevan V., Warden P., Wicke M., Yu Y., Zheng X. TensorFlow: a system for large-scale machine learning // Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation. Savannah, GA, USA: USENIX Association Berkeley, 2016. P. 265–283.

[42] Прикладная статистика. Классификация и снижение размерности / Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д., Под ред. Айвазяна С. А. М.: Финансы и статистика, 1989. 607 с.

[43] Scikit-learn Machine Learning in Python URL: http://scikit-learn.org/stable/index.html (дата обращения: 23.04.2018).

[44] Глубокое обучение для новичков: распознаем изображения с помощью сверточных сетей // Хабрахабр URL: https://habrahabr.ru/company/wunderfund/blog/314872/ (дата обращения: 23.04.2018).